Accurate Detection of Automatically Spun Content via Stylometric Analysis

Usman Shahid
Lahore University of Management Sciences and University of Illinois at Chicago

Shehroze Farooqi
The University of Iowa

Raza Ahmad
Lahore University of Management Sciences

Zubair Shafiq
The University of Iowa

Padmini Srinivasan
The University of Iowa

Fareed Zaffar
Lahore University of Management Sciences

***Abstract*—Spammers use automated content spinning techniques to evade plagiarism detection by search engines. Text spinners help spammers in evading plagiarism detectors by automatically restructuring sentences and replacing words or phrases with their synonyms. Prior work on spun content detection relies on the knowledge about the dictionary used by the text spinning software. In this work, we propose an approach to detect spun content and its seed without needing the text spinner's dictionary. Our key idea is that text spinners introduce stylometric artifacts that can be leveraged for detecting spun documents. We implement and evaluate our proposed approach on a corpus of spun documents that are generated using a popular text spinning software. The results show that our approach can not only accurately detect whether a document is spun but also identify its source (or seed) document — all without needing the dictionary used by the text spinner.**

I. Introduction

Background. The golden rule of Search Engine Optimization (SEO) is that "content is king." High-quality content is crucial to increase search engine ranking that in turn attracts traffic to a website. Top websites invest heavily in content writers to generate high-quality content for them. Spammers create bogus websites that are often loaded with abusive advertisements and malware and they use a variety of black-hat SEO techniques to improve their search engine rankings [20], [24], [26], [33], [34], [42]. A commonly used black-hat SEO technique is to copy high-quality original content from popular websites onto their websites. Search engines try to identify and penalize the rankings of websites that use such plagiarized content [2].

Problem Statement. To evade plagiarism detection, spammers try to "spin" copied content by making it look different without changing its meaning. Spinning can be done manually or automatically. For manual spinning, spammers use crowdturfing websites [43] and black-hat marketplaces [23] to recruit cheap human labor. For automatic spinning, spammers use text spinning software such as The Best Spinner (TBS) [9] and Spinbot [7]. Text spinners automatically restructure sentences and replace words or phrases with their synonyms. Using text spinning software, spammers can create many spun versions of an original article.

Limitations of Prior Art. Plagiarism detection is broadly of two types: (i) extrinsic, where the decision is made by comparing the document to items in a reference document collection [15], [29], [32], [37], [44] and (ii) intrinsic, where the document is assessed in isolation [21], [39], [40]. To the best of our knowledge, there is limited research on plagiarism detection in the context of text spinning. Automatic text spinning has only recently started to receive attention by the research community. In a seminal work, Zhang et al. [46] proposed DSpin to detect spun content based on *immutables*, i.e., words and phrases that are not modified by the text spinner. The authors identify immutables by reverse-engineering the text spinner to gain access to its synonym dictionary. Words and phrases not in the synonym dictionary are immutables. To quantify the similarity between two articles, DSpin computes overlap in immutable words using the Jaccard coefficient. While DSpin accurately detects spun articles, its reliance on knowledge about the synonym dictionary used by the text spinner may not always be tenable. For example, TBS software now uses a cloud-based synonym dictionary to thwart reverse engineering attacks. Moreover, the specific reverse-engineering attack used to identify TBS's synonym dictionary would not work against other spinning software, which are now aware of DSpin and have added new safeguards and countermeasures for evasion.

Proposed Approach. We aim to detect spun content generated using text spinning tools and we aim to do this without relying on knowledge about the text spinner's synonym dictionary. This problem is challenging because sophisticated spinning software offer a variety of configuration parameters to generate a large number of spun versions from a single seed document. To address this challenge, we propose a two step approach: first detect spun content and then identify its seed document. We detect whether a document is spun based on its intrinsic stylometric characteristics. Our key insight is that spinning software introduce lexical artifacts in spun documents that can be leveraged to detect them. Once detected, we use an extrinsic analysis technique to identify its seed — if it is present in a reference document set built from reputed original content sources.

We summarize our key contributions and findings below.

1) We consider a large variety of features for intrinsic spun document detection: basic lexical, vocabulary richness, readability, syntactic as well as n-grams. Additionally, we consider features at different granularities, ranging from word, sentence, paragraph, to the whole document. We also include their summary statistics such as mean and standard deviation as features. Our best results, obtained by using a combination of all features, give a near perfect F1-Score of 99.94%. Feature analysis of our spun document detection classifier indicates that perplexity is the best feature followed by Honore's R Measure. Most importantly, the key features are not ones that can be easily manipulated using automated spinning, hence spammers will not be able to adapt after knowing about our approach.

2) Given a spun document, we also try to find its seed document. When we are certain the seed is present in the reference collection, we identify it with 99.44% accuracy. We also implement a decision tool using sequence alignment scores to decide whether the seed is present in the reference collection. The decision accuracy of this tool varies from 75% to 97% as the percentage of missing seeds varies from 90% to 10%.

II. Text Spinning Workflow

Text spinners are sophisticated tools that allow spammers to evade plagiarism detection. Text spinners can generate multiple copies of an original article by randomly replacing synonyms and changing sentence structure in such a way that the copies remain semantically similar but are syntactically and lexically different. While manual text spinning may generate more readable articles, this approach is clearly cost-prohibitive and unscalable. Thus text spinning software such as The Best Spinner (TBS) [9] and Spinbot [7] which are readily available for automatic spinning are highly attractive to spammers.

It is not sufficient to simply generate a large number of spun articles using a text spinner. An effective spammer also has to evade the various countermeasures used by search engines and other aggregation services. Typically search engines use plagiarism detection tools to detect and penalize web pages and their domains [2], [5]. Spun texts of poor quality are also penalized in search rankings [6]. Given this scenario, we illustrate a realistic text spinning workflow in Figure 1. While we acknowledge that we cannot be completely certain of the process used by spammers, we are confident that they adapt continually to plagiarism detectors and that they understand the importance of content quality.

• A spammer inputs a seed article to a text spinner to generate its spun version. Since the process of spinning an article is non-deterministic, text spinner can generate multiple versions of the same seed article.

• Since text spinner replaces multiple words and phrases randomly, the spun text can become unintelligible. In order to discard spun documents that are overly unintelligible, a spammer would perform automated grammar and readability checks to discard unintelligible articles using off-the-shelf automated language checking tools [3], [4]. Note that some spinning software include built-in language checking tools.

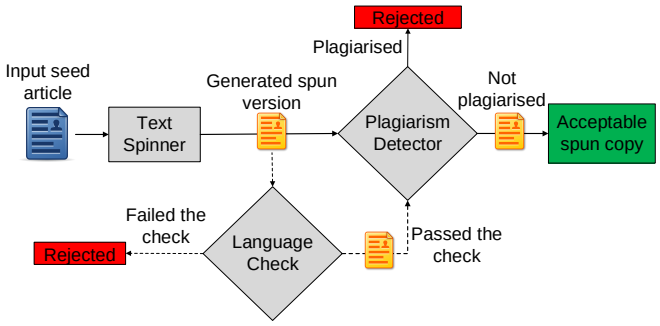


Fig. 1. Workflow typically followed by a spammer to generate a spun version of an article to post on website. Language check is optional.

• The spun article is checked by a plagiarism detector. If the spun article bypasses the plagiarism detector, the spammer posts that article to the website. Otherwise, the spun article is rejected.

III. Data

In this section, we describe the step-by-step creation of a dataset that will be used as a representative sample of legible spun documents which evade plagiarism detection tools.

Text Spinning Tool. There are numerous automatic text spinning tools available for purchase online. These can be identified by querying search engines or searching blackhat forums. Most charge a subscription fee. For example, TheBestSpinner (TBS) [9] and Spinbot [7] respectively cost $47 and $75 per year. We select TBS as the text spinner in our workflow because it is popular and has been studied previously [46].

Seed Article Repository. Spammers need a high-quality, large source of seed articles to generate spun content in bulk quantities. Wikipedia provides a large source of original content on the web. We use a dump of 8,536,467 articles from the English-version of Wikipedia, downloaded in July 2016 as our source of seed articles. From this initial set, we remove all articles containing less than a 100 words[1] leaving us a set (*wiki-100-plus*) of 3,059,052 articles.

Plagiarism Detection Tool. There are several online and offline plagiarism detection tools such as Turnitin [10] and Copyscape [1]. While some offer a free version, most have a premium version with less restricted and advanced plagiarism detection techniques. As part of our text spinning workflow, we use the premium version of Turnitin available to us through an institutional subscription. Turnitin assesses a given document against a reference collection which can be online (e.g., Wikipedia, technical journals) or a local custom

[1]Wikipedia articles with less than 100 words are often placeholders or contain image captions.

Parameter	Description	Possible Values
Dictionary	Use cloud based dictionary or local one	Cloud/Local
Rewrite Sentences	Whether to rewrite sentences before spinning	True/False
Phrase Only	Replace "both" words and phrases or just "phrase only"	Both/Phrase Only
Max Synonyms	Maximum number of replacements per word/phrase	1,2,3 ...
Replacement Frequency	Replace every k^{th} word	1,2,3,4
Autospin Inside	Whether to perform nested spinning or not	True/False

TABLE I
PARAMETERS OF THE BEST SPINNER (TBS)

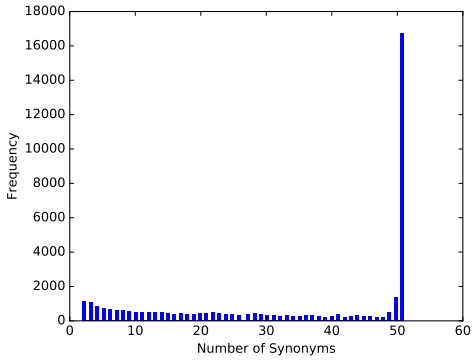


Fig. 2. Histogram of the number of available synonym replacements in the Spintax of 100 randomly sampled articles spun using Max Synonyms=1000.

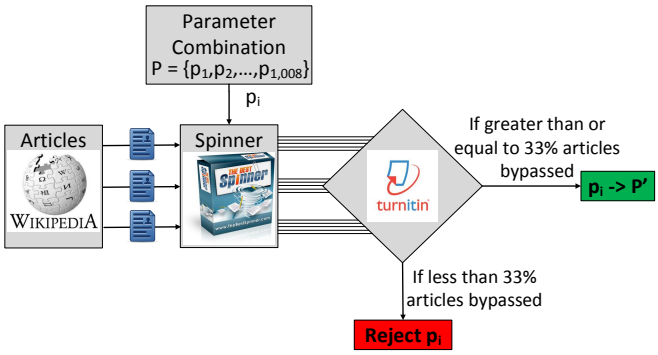


Fig. 3. Workflow to identify TBS parameter combinations bypassing Turnitin. P' represents the set of parameters that bypass Turnitin.

article repository. Turnitin provides a similarity score between 0-100, computed using a proprietary algorithm; low scores indicate original content and high scores indicate potential plagiarism. We make the problem challenging by using a low similarity threshold of 20%. Lower similarity threshold means that successfully spun articles would be more challenging for search engines to detect. Interestingly, as we show later, thousands of spun documents slip by Turnitin even at this low threshold.

TBS Spinner Parameters. TBS offers a number of parameters which produce different variations of spun content (see Table I). These tunable parameters specify various aspects of the spinning process including but not limited to the use of a local or cloud based dictionary, the replacement frequency, and granularity of spinning. The spinner produces an intermediate output called the 'spintax' which groups the original words/phrases in an article with a set of synonym word and phrase replacement choices based on a dictionary. TBS produces multiple spun copies of an input article by randomly replacing original words/phrases with their replacement candidates from the spintax. As an example, given the sentence 'Writing articles is fun', its spintax could be '{Writing | Creating} {articles | stories} is {fun | enjoyable}'.

We experimentally determine that TBS generates at most 50 synonyms replacement candidates for a word/phrase (see Figure 2). We further sample this range by selecting 17 different values between 2 and 50, boundaries inclusive. All the other parameters are shown in Table I have a finite range. The resultant total number of TBS parameter combinations is 1,088.

Selecting TBS Spinner Parameter Combinations. Next, we conduct a pilot experiment to study which TBS parameter combinations are more likely to bypass Turnitin. Following the workflow in Figure 3, for each of the 1,088 TBS parameter combination, we randomly select 3 seed articles from *wiki-100-plus* without replacement and spin each seed article 5 times to generate 15 spun articles. If at least 5 out of 15 spun articles for a parameter combination have Turnitin similarity score at or below 20%, i.e., bypass Turnitin detection, we consider that parameter combination viable for large-scale dataset generation.

Figure 4 plots the percentage of TBS parameter combinations that bypass Turnitin as a function of different parameters. We want to analyze the impact of each parameter in bypassing detection by Turnitin. Different values of *dictionary*, *autospin inside*, and *rewrite sentences* equally bypass detection by Turnitin. However, the values of *phrase only*, *replacement frequency*, and *max synonyms* differentially impact our ability to bypass Turnitin. We are much more likely to bypass Turnitin when we decide to replace both words and phrases and when we replace every word/phrase with its synonym. For *max synonyms*, we note that all values larger than 2 bypass Turnitin with the same likelihood. Based on these pilot results, we set *phrase only* to replace both words and phrases, *replacement frequency* to 1, and *max synonyms* to larger than 2 for generating our dataset. We use all values for the other parameters. Using this criterion, we select a total of 128 TBS parameter combinations that are likely to bypass Turnitin.

Language Check. As discussed earlier, search engines may reject or at least lower the rank of poor quality web pages. Thus a typical spammer's workflow might include a language check. We emulate this step using a language filter implemented with an n-gram language model. At a high level, an n-gram language model indicates the likelihood of observing an input document with respect to the n-gram distribution

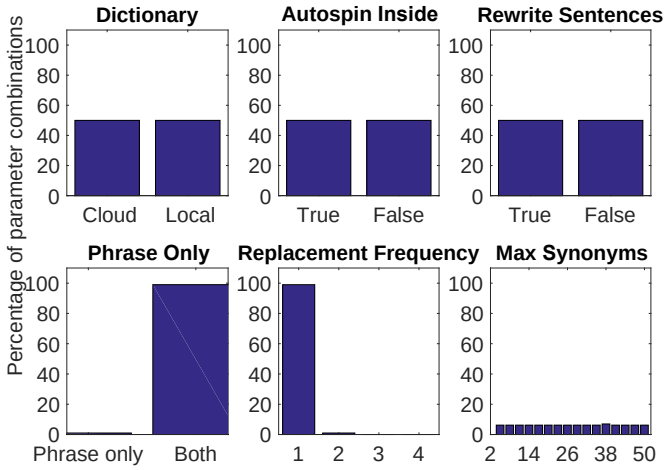


Fig. 4. Percentage of 128 parameter-combinations that bypass Turnitin is given on y-axis and x-axis indicates the possible values of each parameter.

of a reference corpus. This is typically represented using perplexity [18]. High perplexity scores indicate that the input document contains "unexpected" language or n-grams. Lower scores indicate that the input document contains more expected n-gram patterns. We want to assess the perplexity of spun documents against a reference corpus of non-spun documents and reject those with high perplexity as detectable. While this approach does not directly address grammatical issues, it does gauge an input document against a reference corpus exhibiting "acceptable" writing style.

We train an n-gram language model using the open-source SRILM toolkit [8] on a subset of *wiki-100-plus*. We sample 100,000 documents without replacement for training a 4-gram model with modified Kneser-Ney [19] smoothing and plot the distribution of perplexity scores. We sample 40,960 documents from *wiki-100-plus* without replacement, while ensuring that no document which was previously used in training is selected in the sample. We compute perplexity score and note that the median and 99 percentile values of perplexity scores are 194 and 597, respectively, as shown in Figure 5. We select the threshold for the perplexity score at the higher end and reject a spun article if its perplexity score is higher than 597.

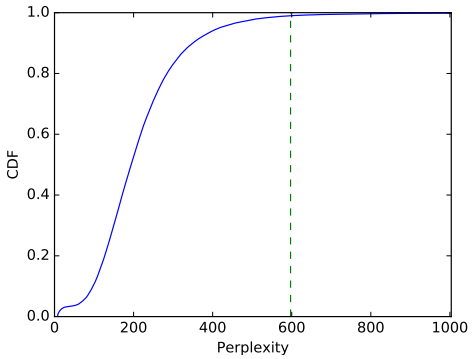


Fig. 5. Distribution of perplexity for non-spun documents. 99% of non-spun content lies below the threshold of 597 (green dotted line).

Dataset Generation. We are now ready to generate our dataset of spun articles. For each of the 128 TBS parameter combinations selected in our pilot experiment, we randomly select 320 seeds from *wiki-100-plus* without replacement and generate 5 spun articles per seed. This yields a set of 204,800 spun articles. Filtering out those with perplexity score $>= 597$, we are left with 14,747 (6.7%) spun articles. Of these, 13,807 (93.6%) bypass detection by Turnitin, i.e., their similarity score is less than 20%. These 13,807 spun articles that have bypassed the language model and Turnitin from our set of spun documents are called *spun-set*. Each has its corresponding seed document in *wiki-100-plus*. Moreover, we build a dataset for our classifier experiments using *spun-set* and a random sample of 138,070 articles from *wiki-100-plus* dataset. We ensure that this random sample does not contain any seeds used for the generation of *spun-set*. We call this dataset *spun-classification-set*. In total, *spun-classification-set* contains 151,877 documents. To reflect realistic class imbalance, the number of non-spun articles are 10 times more than the number of spun articles.

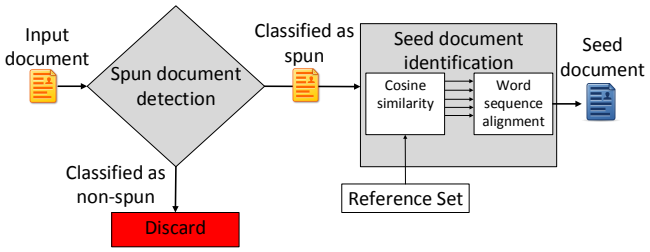


Fig. 6. Our two step approach for spun document detection and seed document identification

IV. Proposed Approach

Overview. As illustrated in Figure 6, our proposed approach consists of two distinct steps: *spun document detection* and *seed document identification*. In the first step, we classify an input document as spun or non-spun. If classified as spun then in the second step we identify the seed document used to generate the spun document. Our approach is aimed particularly at search engines that crawl and index trillions of web pages. A search engine may obviously take advantage of their very large document collections and conduct pairwise comparisons with a new document to see if it is spun or not. This approach, however, is not scalable because they would have to perform trillions of comparisons and that too for every new document crawled. Our two-step approach addresses this scalability challenge by first classifying whether a new document is spun or not using only the document's *intrinsic features*. If and only if the new document is identified as spun, we move to the second step of identifying its seed by comparing against the indexed documents. The efficiency gains are made because our approach limits the pairwise comparisons to only those documents that are detected as spun. This second step may also be optional if the search

engine/website is not interested in finding the seed. In sum, for spun document detection we use a classifier trained on a wide variety of features intrinsic to documents. For seed document identification, we use an extrinsic approach where we measure cosine similarity coupled with word sequence alignment score of spun documents against a set of reference documents.

A. Spun Document Detection

In spun document detection, we determine whether an incoming document has been generated through an automatic spinning process. We use a classifier trained on the intrinsic features of documents. Our key insight is that we are essentially addressing an authorship problem where the author of interest is a text spinning tool. In preliminary analysis, we compared seed documents and their spun counterparts looking for features that differentiate the two. We observe that style, language, grammatical constructs, and certain linguistic expressions in spun documents deviate from a human author because spinning software introduce artifacts in their output which are specific to a text spinner. Therefore, we can train a machine learning classifier on a wide variety of stylometric features to detect spun documents.

1. N-grams	**Granularity**
1-gram, 2-gram, 3-gram	Document
2. Basic Lexical Features	**Granularity**
Character Count [47]	Word, Paragraph
Word Count [40]	Sentence, Paragraph
Syllable Count [40]	Word, Paragraph
Sentence Count [47]	Paragraph
Function Words Percentage [38]	Paragraph, Document
Punctuation Characters Percentage [47]	Paragraph, Document
Special Characters Percentage [47]	Paragraph, Document
Word Unigram Percentage [38], [40]	Paragraph, Document
3. Vocabulary Richness Features	**Granularity**
Word Frequency Class [21]	Word, Paragraph
Type-token Ratio [38]	Paragraph, Document
Sichel's S Measure [14]	Paragraph, Document
Brunet's W measure [14]	Paragraph, Document
Yule's K Characteristic [40]	Paragraph, Document
Honore's R Measure [40]	Paragraph, Document
Simpson's Index [27]	Paragraph, Document
Shannon Entropy [27]	Paragraph, Document
4. Readability Features	**Granularity**
Flesch-Kincaid Grade Level [40]	Paragraph, Document
Flesch Reading Ease [40]	Paragraph, Document
Gunning Fog Index [40]	Paragraph, Document
Dale Chall Readability Formula [40]	Paragraph, Document
5. Syntactic Features	**Granularity**
Syntactic Complexity [41]	Paragraph, Document
POS Trigrams Percentage [40]	Paragraph, Document
Vowel Trigram Percentage [40]	Paragraph, Document
6. Perplexity	**Granularity**
Perplexity Score	Document

TABLE II
LIST OF STYLOMETRIC FEATURES AND CATEGORIES FOR SPUN DOCUMENT DETECTION.

Stylometric Features. A large variety of stylometric features have been studied for authorship attribution and plagiarism detection [14], [22], [38], [40]. While conceptually simpler, n-gram features are also extensively used in authorship attribution research [28]. Table II lists 25 different stylometric features which can be broadly categorized into six different types.

1) N-gram word features: These are standard features capturing frequency distribution of word instances and sequences. We explore 1-gram, 2-gram and 3-gram features and their different combinations. We limit to top 10,000 n-grams for n=1,2,3 based on frequency to reduce dimensionality.
2) Basic lexical features: These are normalized counts for different kinds of lexical constructs. For example, *syllable count* measures the frequency of syllables in a passage.
3) Vocabulary richness features: These evaluate vocabulary usage. For example, *Honore's R measure* considers the proportion of words used only once.
4) Syntactic features: These capture various syntactic structures at the character and word levels. For example, *vowelness trigram* counts different permutations of vowels and consonants in a character trigram.
5) Readability features: These measure the difficulty or ease in comprehending a text passage through standard metrics as *Gunning fog index* and *Dale-Chall readability formula*. Both consider sentence length and the number of complex words.
6) Perplexity: We seek to determine the value of perplexity as a stand alone feature in order to see if it has additional value even after being used as part of our language model filter.

Categories 2 through 5 in Table II generate several features based on granularity level and configurations. For example, the feature *word count* is defined both at sentence- and paragraph-level granularities. Table II also shows the granularity levels. Note this does not apply to n-gram features or perplexity which are computed only at the document level. Features can also be specified with different configurations. For example, *word unigram percentage* measures the occurrence of a particular word as a percentage of total words at paragraph and document level for 6 different words (such as 'from'), thus yielding a total of 12 different features at the paragraph- and document-level. For most feature categories, we compute the following 8 summary statistics: *min, max, median, mean, standard deviation, skewness, kurtosis*, and *entropy*. These form distinct features as well. For example, we compute these 8 statistical features for "sentence count" assessed with paragraph level granularity. In sum, we define a total of 415 features. Note again that the n-gram features are simpler and do not have these nuances.

Classifier. We train a supervised SVM classifier to decide if a document is spun or not based on the aforementioned features. SVMs have been successfully used in prior literature with stylometric features for plagiarism detection and authorship attribution [30], [40]. It is noteworthy that the classifier has

to handle an imbalanced dataset where the number of non-spun articles is at least an order of magnitude more than the number of spun articles to reflect real-world class imbalance. The documents detected as spun are then passed to the next step for identifying their seed documents.

B. Seed Document Identification

Our next goal is to identify seeds for the documents classified as spun. We do this by comparing each spun document against a set of reference documents. Search engines can use various original content sources, such as news websites, popular blogs, and Wikipedia, as the reference set. It is noteworthy that the seed document may not be present in the reference set and we handle such cases.

We use two different methods to find the seed of a spun document. First, we use cosine similarity to identify the five most similar documents from the reference set. We limit this to a small set of five documents in case the search engine administrator opts for manual assessment at this point. For cosine similarity, each document pair (the spun document and the reference document) is represented by a vector of *tf-idf* scores. We then rank the reference documents based on their cosine similarity to the spun document which gives the top-5 ranked seed candidates. We expect the seed to be present in the top-5 candidates - if it is in the reference collection.

Second, we use a more sophisticated word sequence alignment technique to decide if the seed is present in the top-5 seed candidates. Since the spinner does not replace certain "immutable" words, word sequence alignment can capture similarity based on the ordering of such immutable words. Note we do not need to know what these words are, but simply rely on the observation that immutables will exist. The limitation with cosine similarity is that it treats the document as a bag of words and so ignores word sequence information. Instead, here we use sequence alignment score which is the ratio of words aligned in the spun and non-spun pair of documents being compared to the total number of words in the spun document. Table III illustrates an example of immutable words of a seed and its spun version. The total number of words aligned is 14 and the total number of words in the spun document are 20 giving an alignment score of $14/20 = 0.7$.

Seed	Spun
If you are interested in **the products you only** need **to research and** identify **the products that meet your** ideals.	**If you** should be contemplating **the products you only** have **to research and** determine **the products that meet your** beliefs.

TABLE III
COMPARISON BETWEEN SEED AND SPUN
SHOWING SAME SEQUENCE OF COMMON WORDS

In sum, we use cosine similarity to find the five most similar documents to a spun document and we expect the seed to be contained in this set of five. However, since the seed may be absent altogether from the reference collection we also use sequence alignment scores to decide whether the seed is present. If the sequence alignment score for a document in the top 5 candidates is high enough then we declare that the seed is present otherwise it is absent. We provide more details about this threshold in the next section.

V. EXPERIMENTS AND RESULTS

A. Spun Document Detection

We begin with our SVM-based spun document classification experiments. These are done using *spun-classification-set* described in Section III which has 151,877 documents, of which 13,807 are spun. We use Python's `scikit-learn` implementation of SVM.

Finding Optimal Parameters. For each experiment, defined by a particular feature combination, we find optimal parameter values using a stratified random sample of 41,877 documents from *spun-classification-set*. This dataset is called *parameter-selection-set*. The SVM hyper-parameters: C, γ, kernel type are optimized for macro-averaged F1-measure. We do this via grid search using a stratified 5-fold cross-validation design. All siblings (documents spun from the same seed) are placed in the same fold to avoid risk of simplifying the problem. We have explored linear and RBF kernels in the grid search and values for C and γ are searched within $\{1, 10, 100, 1000\}$ and $\{0.1, 0.01, 0.001, 0.0001\}$ respectively. The RBF kernel provides the best results for all experiments. C is either 10 or 100 and γ is either 0.001 or 0.0001 for n-gram feature combinations. For other stylometric features, C and γ are generally below 100 and 0.01 respectively.

Classification Results. The optimized classifiers are tested on the 110,000 documents remaining in *spun-classification-set* after removing the data used for parameter selection (*parameter-selection-set*). Again, we use stratified 5-fold cross validation while placing sibling spun documents in the same fold.

Table IV presents results organized by feature set. The first key observation is that the best results are close to perfect. The combination of all feature categories gives micro-averaged precision, recall, and F1 of 100%, 99.89%, and 99.94%, respectively, over the 5 folds. We obtain high accuracy despite the fact that spun documents are not detected either by Turnitin or by the language model. Moreover, our dataset is heavily skewed towards non-spun documents.

Second, we note that n-grams surprisingly perform better than most of the other feature categories. Its best results are about 1% better in terms of F1 than the best with remaining stylometry features. While numerically noticeable, this is not an appreciable difference. Among other stylometric categories, vocabulary richness is the best single category classifier (F1 = 95.35%) while readability performs the worst (F1=12.76%). Ablation analysis, where we exclude one category at a time along with n-grams, indicates that the stylometric features capture overlapping characteristics. For example, removing vocabulary richness (the best single category classifier) alone

Feature Type	Number of Features	Precision (%)	Recall (%)	F1 (%)
1-gram	10000	99.79	76.45	86.57
2-gram	10000	99.96	99.76	99.86
3-gram	10000	99.87	97.73	98.79
1,2-gram	20000	99.95	99.86	99.90
2,3-gram	20000	99.97	99.82	99.89
1,3-gram	20000	99.91	99.82	99.86
1,2,3-gram	30000	99.97	99.86	99.91
All Except N-gram	415	98.99	98.54	98.77
All Features	30415	100.00	99.89	**99.94**
Basic Lexical*	278	98.37	97.99	98.18
Basic Lexcial**	137	89.07	80.05	84.32
Readability*	379	98.96	98.59	98.77
Readability**	36	75.35	6.97	12.76
Vocabulary Richness*	336	96.71	96.58	96.65
Vocabulary Richness**	79	96.42	94.31	95.35
Syntactic*	253	99.11	98.73	98.92
Syntactic**	162	58.72	41.60	48.70
Perplexity*	414	98.27	96.59	97.42
Perplexity**	1	52.05	90.23	66.02

TABLE IV
RESULTS FOR SPUN DOCUMENT DETECTION WITH DIFFERENT STYLOMETRIC FEATURE COMBINATIONS FOR *spun-classification-set*.
*: THE CATEGORY AND THE N-GRAM FEATURES ARE EXCLUDED FROM THE FEATURE SET.
**: ONLY FEATURES OF THIS CATEGORY ARE USED.

does not do much harm. Similarly, removing the next best, basic lexical feature set, does not significantly degrade accuracy.

The strong and about equal performance of n-grams deserves additional attention. A key point is that n-gram feature sets are more than the 415 stylometry features by several orders of magnitude. N-grams are not as informative to a human as compared to stylometric features. Most crucially, the n-gram feature space is also determined by the training data used. Therefore, we postulate that as the tested document deviates in topic from the training set, accuracy may decrease. Whereas we expect stylometric features to function *relatively* independent of training data.

Analysis of Select Stylometric Features The benefit of stylometric features is that they provide insights into the way text spinners work and the features that may differentiate spun from non-spun. Due to space constraints, we analyze only a select subset of features, mostly those that performed well when used in isolation. Each plot in Figure 7 is a distribution for a feature showing the distinction between spun and non-spun documents.

Vocabulary richness is a category that fared best when used as the sole feature set. For this we plot the distribution of Honore's R measure paragraph mean in Figure 7(a). As expected, there is a clear distinction between spun and non-spun documents. The distribution of Shannon entropy at document level in Figure 7(b) also displays similar distinguishing nature. These observations corroborate author attribution research which highly values vocabulary richness features [22].

Basic lexical feature category is the second best when used in isolation. For this we plot the distribution of character counts entropy in Figure 7(c) and syllable counts entropy in Figure 7(d) at word level. We see that for spun documents, entropy in both these feature classes is quite static within a very small range of values. However, for non-spun, the spectrum of values is comparatively big, ranging from 3 to 10. Despite being fairly basic in nature, these features are able to measure subtleties in lexical and contextual information [38].

Perplexity is the third best performer when used in isolation. Figure 7(e) indicates that perplexity of spun documents is higher as compared to most non-spun documents. This is likely because when the text spinner replaces words and phrases randomly with their synonyms, the generated spun content becomes relatively incoherent and disordered. But yet since the separation is not as strong as with vocabulary richness and basic lexical features, its performance in isolation is the weakest of the three categories.

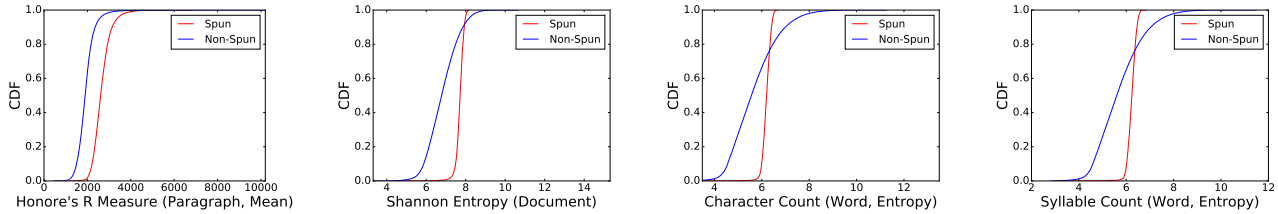
We also examine readability category with a plot for paragraph entropy of Flesch-Kincaid Grade in Figure 7(f). While it has relatively higher values for spun compared with non-spun, the separation is even less clear than with the previous features leading to the lowest accuracy. Similarly syntactic complexity category features which do not fare well in the classifier experiments also do not discriminate very well between spun and non-spun, as indicated by Figure 7(g). One possible reason for non-discrimination of these categories is that we only consider the spun content that evades the language model, possibly making it syntactically closer to non-spun content and increasing the general readability.

Finally, we also analyze features using information gain which ranges from 0 to 1. Perplexity stands out among all the features with the highest score of 0.27. The next best nine positions are taken by features in vocabulary richness. For example, two specific features derived from Honore's R Measure occupy the second and third spot with information gain scores of 0.15 and 0.14. Unsurprisingly, all readability and syntactic complexity features have low information gain scores that are less than 0.05. As we have discussed, these features are not able to accurately distinguish between spun and non-spun documents.

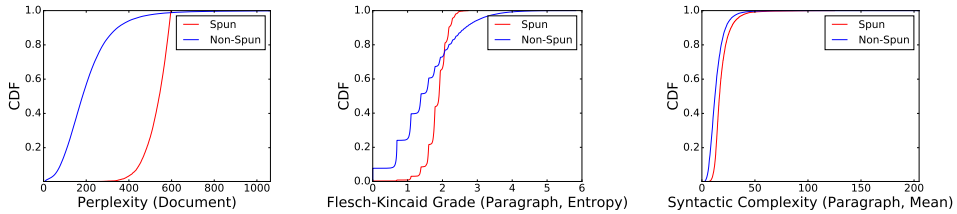
B. Seed Document Identification

We exclude all the non-spun documents used in *spun-classification-set* from *wiki-100-plus* which leaves us with 2,920,982 documents. We call this dataset *reference-set*. Note that all seed articles which were used to generate *spun-set* are present in *reference-set*. We use *reference-set* to run experiments for seed document identification. We present two sets of results. The first set of results corresponds to all seeds being present in the *reference-set*. The second set of results corresponds to an experiment where we remove seeds and evaluate our approach.

1) Results (Seeds Present): As described earlier, we use cosine similarity between *tf-idf* representations to obtain top-5 documents from *reference-set* with the expectation that the seed is in this ranked set. Table V provides the results. We observe that cosine similarity accurately identifies the seed



(a) Honore's R measure paragraph mean (b) Shannon Entropy document (c) Character count word entropy (d) Syllable count word entropy

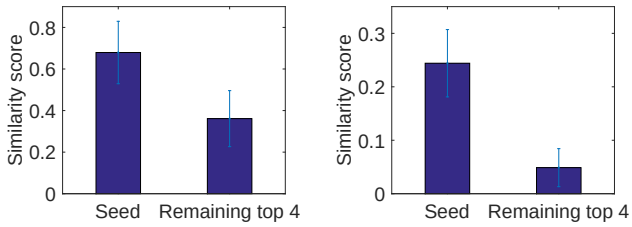


(e) Perplexity (f) Flesch-Kincaid Grade paragraph entropy (g) Syntactic Complexity Paragraph Mean

Fig. 7. Distribution of spun and non-spun documents for a select subset of stylometric features across different categories excluding n-gram.

Rank	Documents (%)
1	99.44
2	99.77
3	99.84
4	99.89
5	99.91

TABLE V
PERCENTAGE OF SEED DOCUMENTS FOUND AT OR ABOVE THE SPECIFIED RANK USING COSINE SIMILARITY.



(a) Cosine similarity (b) Word Sequence alignment

Fig. 8. For each spun document, we select top 5 documents based on cosine similarity from the *reference-set*. Figure (a) show average cosine similarity of spun documents against seed and non seed documents in these top 5 documents. Figure (b) shows average word sequence alignment score of spun documents against seed and non seed documents in these top 5 documents. Word sequence alignment scores differentiate seed from non seed more effectively than cosine similarity.

document in the top-5 with an accuracy of 99.91%. Moreover, the seed document is ranked at the first position 99.44 % of the times. These close to perfect results with cosine similarity are best case results when we have all the seeds in the *reference-set*.

2) Results (Seeds Absent): While cosine does extremely well in retrieving the seed when present, it has a handicap when it comes to *deciding* whether the seed is present in the *reference-set*. As indicated in Section IV, this handicap is because cosine similarity does not consider word sequence in the documents being compared. A better alternative we introduced earlier is word sequence alignment and this advantage is demonstrated in Figure 8. The figure shows the average cosine similarity score and average word sequence alignment score for the spun document when compared with its seed and when compared with the remaining documents ranked in the top 5 positions by cosine similarity. The difference is much higher for word sequence alignment in comparison to cosine similarity. This motivates our choice of word sequence alignment.

If the sequence alignment score between the seed and any of the top-5 documents returned by the cosine similarity step is above a threshold, we declare that the seed is present. Thus, our first goal is to find a suitable threshold for the word sequence alignment score.

Identifying Threshold. We make a 70%–30% split of the *spun-set* dataset to get the training and testing set, respectively. We use the training set to identify the best threshold using a bootstrapping approach. Essentially, we bootstrap a random sample of size equal to the training set by sampling with replacement and then picking a threshold value such that 99% of the spun-seed sequence alignment scores are higher. We do this for a 100 different samples and get the median threshold value of 0.08781. We use this as our final threshold. Thus, the decision criteria is as follows: if the sequence alignment score for any of the top-5 documents returned by sequence alignment is higher than 0.08781, we state that the seed is present otherwise we state that it is not present.

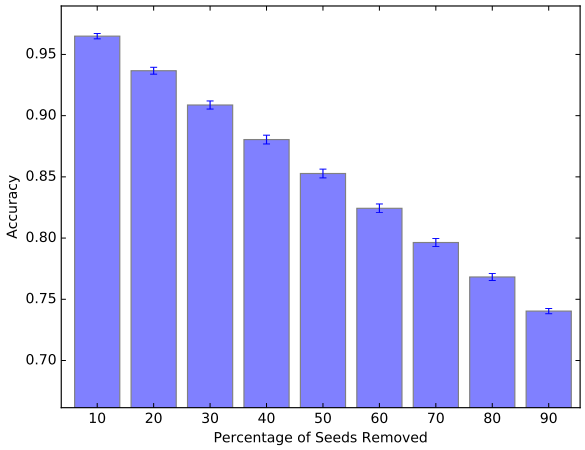


Fig. 9. Mean accuracy of word sequence based decision tool. The x-axis indicates the percentage of seeds removed from the *reference-set*.

Evaluation of Decision Criteria. Figure 9 shows accuracy achieved by this decision strategy based on the sequence alignment score and threshold. This experiment was done on the 30% test set derived from *spun-set* dataset after finding the top-5 documents from *reference-set* using cosine similarity. The y-axis reports accuracy of the decision. The x-axis varies the percentage of spun articles whose seeds are missing in the *reference-set*. For example, when x is 10 it means that 10% of the seeds are missing. We add robustness to these results by repeating the experiments 1000 times (for each value on the x-axis) and reporting standard deviation of accuracy using error bars. The accuracy of our decision strategy ranges between 75% and 97% as shown in Figure 9. We observe that accuracy decreases as the percentage of spun articles whose seeds are missing increases. A primary reason for this decrease is that several Wikipedia articles used in *reference-set* follow a similar template with very similar word sequences. In the absence of a seed document, the sequence similarity score of other candidate seeds that follow a similar template results in the threshold being crossed, and an incorrect decision being made. Despite this being a challenging problem, we note that our average accuracy in the worst case is still 75%.

VI. Related Work

Web spam has been around almost as far back in time as the web itself. Content spamming, link spamming, content hiding, and cloaking are few of the techniques used by spammers [26]. Counter strategies have also been developed including methods that analyze content, cross reference multiple web pages, traverse link structures and explore statistical properties of websites [16], [20], [24], [33]. However, we focus on tracking the large-scale spam generated using text spinning tools.

Since it is common for spammers to plagiarize existing content, plagiarism detection is an active area of research and it has two major lines of thread, namely extrinsic [12], [17], [35], [36], [44], [45] and intrinsic detection [13], [21], [39], [40]. However, detecting spam generated using text spinners has barely received attention. DSpin [46] is the one exception which explores an extrinsic strategy comparing a new document with items in a collection. Moreover, they compute document similarity with respect to a set of 'immutable' words - words not in the text spinner's dictionary. In essence, they reverse-engineered TBS to obtain its dictionary and then considered words not present in it as immutables. This technique is limited firstly because extrinsic techniques require a large reference corpus to compare against. Thus such tools are not useful to web sites with small reference collections. Moreover, while the likelihood of catching spam improves with bigger collections, these comparisons - done for each new document - become increasingly expensive computationally. The second major limitation lies in the dependence on the text spinner's synonym dictionary. It may not always be possible to reverse-engineer the synonym dictionary used by text spinners. Not surprisingly TBS has adapted [11]; its synonym dictionary is now cloud-based which makes it impossible to identify the immutable words. In contrast, our approach is intrinsic in nature and independent of the spinner's dictionary, making it computationally cheaper and more widely usable. Since we cannot obtain the cloud-based synonym dictionary used by TBS now, we are unable to replicate results from DSpin on our dataset and compare our approach against it.

Prior literature on plagiarism detection [21], [22], [25], [31] and authorship attribution [27], [30], [38], [40] shape the rich set of stylometric features we use. While our focus is on plagiarism, we view our problem as a specific version of the author attribution problem; the difference is that our author of interest is the text spinning tool itself. In a seminal work, Holmes outlined about 16 kinds of authorship characteristics [27] including passage lengths and vocabulary distributions. Koppel and Schler suggest appropriate authorship attribution techniques for different problem scenarios [30]. Stamatatos extensively surveys modern authorship attribution methods [38] and also categorized stylometric features based on their efficacy. Likewise, in intrinsic plagiarism detection, Stein et al. categorize and rank stylometric features on the basis of their discriminative power [40]. Similarly, Eissen et al. also present various linguistic features for quantifying writing style of different parts of a document [22]. Stamatatos introduces the concept of style changes across a document using character n-gram profiles [39]. We make extensive use of all of these features.

VII. Conclusion

In this paper, we presented an approach to accurately detect spun documents and their source without relying on any knowledge about the synonym dictionary used by the spinning software. To this end, we were able to leverage stylometric artifacts introduced in spun documents by the spinning software. Our key contributions are as follows. First, we generated a dataset of spun documents using a popular spinning tool that bypassed plagiarism detection as well as

a language check. Second, we proposed a two-step approach to first detect spun documents and then identify their seed documents. Third, we implemented and rigorously evaluated our proposed approach on a large dataset of spun and non-spun documents. The results showed that our approach can detect spun documents with 99.94% F-score and identify source seed document with 99.44% accuracy. In future, we plan to evaluate the effectiveness of our proposed approach on different text spinners.

Acknowledgements

This work is supported in part by the National Science Foundation under grant number CNS-1715152.

REFERENCES

[1] "Copyscape Plagiarism Checker - Duplicate Content Detection Software," http://www.copyscape.com.
[2] "Duplicate content," https://support.google.com/webmasters/answer/66359.
[3] "grammar-check 1.3.1," https://pypi.python.org/pypi/grammar-check/1.3.1.
[4] "language-check 1.0," https://pypi.python.org/pypi/language-check.
[5] "Little or no original content," https://support.google.com/webmasters/answer/66361.
[6] "Search Engine Optimization Starter Guide," https://static.googleusercontent.com/media/www.google.com/en//webmasters/docs/search-engine-optimization-starter-guide.pdf.
[7] "Spinbot - Article Spinning, Text Rewriting, Content Creation Tool." https://spinbot.com/.
[8] "SRILM - The SRI Language Modeling Toolkit," http://www.speech.sri.com/projects/srilm/.
[9] "The Best Spinner - the most powerful article spinner software." http://thebestspinner.com/.
[10] "Turnitin - Technology to Improve Student Writing," http://turnitin.com.
[11] "The Best Spinner version 3.5 is out: Auto-Sentence Rewriting and Cloud Thesaurus," http://jlforums.com/tbs-version-3-beta/the-best-spinner-version-3-5-is-out-auto-sentence-rewriting-and-cloud-thesaurus, 2015.
[12] S. Alzahrani and N. Salim, "Fuzzy semantic-based string similarity for extrinsic plagiarism detection," in *Notebook Papers of Conference and Labs of the Evaluation Forum (CLEF) Labs and Workshops*, 2010.
[13] S. Alzahrani, N. Salim, and A. Abraham, "Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods," *IEEE Transactions on Systems, Man, And Cybernetics*, 2002.
[14] R. H. Baayen, A. Lüdeling, and M. Kytö, "Corpus linguistics: An international handbook," vol. 2, 2008.
[15] A. Barron-Cedeno and P. Rosso, "On Automatic Plagiarism Detection Based on n-Grams Comparison," in *European Conference on Information Retrieval (ECIR)*, 2009.
[16] A. A. Benczur, K. Csalogany, T. Sarlos, and M. Uher, "Spamrank–fully automatic link spam detection work in progress," in *International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.
[17] S. Brin, J. Davis, and H. Garcia-Molina, "Copy Detection Mechanisms for Digital Documents," in *ACM SIGMOD*, 1995.
[18] P. Brown, P. deSouza, R. Mercer, V. D. Pietra, and J. C. Lai, "Class-based N-gram Models of Natural Language," *Computational Linguistics*, 1992.
[19] S. Chen and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling," in *Association for Computational Linguistics (ACL)*, 1996.
[20] F. Dennis, M. Manasse, and M. Najork, "Detecting phrase-level duplication on the world wide web," in *ACM SIGIR*, 2005.
[21] S. Eissen and B. Stein, "Intrinsic plagiarism detection," in *European Conference on Information Retrieval (ECIR)*, 2006.
[22] S. Eissen, B. Stein, and M. Kulig, "Plagiarism detection without reference collections," *Journal of the Association for Information Science and Technology (JASIST)*, 2007.
[23] S. Farooqi, M. Ikram, E. D. Cristofaro, A. Friedman, G. Jourjon, M. Kaafar, Z. Shafiq, and F. Zaffar, " Characterizing Key Stakeholders in an Online Black-Hat Marketplace," in *IEEE/APWG Symposium on Electronic Crime Research (eCrime)*, 2017.
[24] D. Fetterly, M. Manasse, and M. Najork, "Spam, Damn Spam, and Statistics," in *ACM SIGMOD/PODS*, 2004.
[25] S. Gruner and S. Naven, "Tool support for plagiarism detection in text documents," in *ACM Symposium on Applied Computing (SAC)*, 2005.
[26] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.
[27] D. Holmes, "Authorship attribution," *Computers and the Humanities*, 1994.
[28] V. Keselj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for authorship attribution," in *Pacific Association for Computational Linguistics (PACLING)*, 2003.
[29] D. Khmelev and W. Teahan, "A repetition based measure for verification of text collections and for text categorization," in *ACM SIGIR*, 2003.
[30] M. Koppel, J. Schler, and S. Argamon, "Computational methods in authorship attribution," *Journal of the Association for Information Science and Technology (JASIST)*, 2009.
[31] W.-Y. Lin, N. Peng, C.-C. Yen, and S. de Lin, "Online plagiarism detection through exploiting lexical, syntactic, and semantic information," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2012.
[32] K. Monostori, A. Zaslavsky, and H. Schmidt, "Document overlap detection system for distributed digital libraries." in *ACM Digital Libraries (DL)*, 2000.
[33] M. Najrok, "Detecting quilted web pages at scale," in *ACM SIGIR*, 2012.
[34] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting Spam Web Pages through Content Analysis," in *ACM WWW*, 2006.
[35] A. H. Osman, N. Salim, and A. Abuobieda, "Survey of text plagiarism detection," *Computer Engineering and Applications Journal (ComEngApp)*, 2012.
[36] N. Shivakumar and H. Garcia-Molina, "SCAM: A copy detection mechanism for digital documents," in *Theory and Practice of Digital Libraries*, 1995.
[37] P. Shrestha and T. Solorio, "Using a Variety of n-Grams for the Detection of Different Kinds of Plagiarism," in *Notebook for PAN at Conference and Labs of the Evaluation Forum (CLEF)*, 2013.
[38] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for information Science and Technology (JASIST)*, 2009.
[39] ——, "Intrinsic Plagiarism Detection Using Character n-gram Profiles ," in *Spanish Society for Natural Language Processing (SEPLN). Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN)*, 2009.
[40] B. Stein, N. Lipka, and P. Prettenhofer, "Intrinsic plagiarism analysis," *Language Resources and Evaluation*, 2011.
[41] B. Szmrecsanyi, "On operationalizing syntactic complexity," in *International Conference on Textual Data Statistical Analysis*, 2004.
[42] D. Wang, S. Savage, and G. Voelker, "Juice: A Longitudinal Study of an SEO Botnet," in *Network and Distributed System Security Symposium (NDSS)*, 2013.
[43] G. Wang, C. Wilson, X. Zhao, Y. Zhu, M. Mohanlal, H. Zheng, and B. Zhao, "Serf and Turf: Crowdturfing for Fun and Profit ," in *ACM WWW*, 2012.
[44] R. Yerra and Y.-K. Ng, "A Sentence-Based Copy Detection Approach for Web Documents," in *Fuzzy systems and knowledge discovery (FSKD)*, 2005.
[45] M. Zechner, M. Muhr, R. Kern, and M. Granitzer, "External and intrinsic plagiarism detection using vector space models," in *Spanish Society for Natural Language Processing (SEPLN). Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN)*, 2009.
[46] Q. Zhang, D. Wang, and G. Voelker, "DSpin: Detecting Automatically Spun Content on the Web," in *Network and Distributed System Security Symposium (NDSS)*, 2014.
[47] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *Journal of the American society for information science and technology (JASIST)*, 2006.