

Filtering artificial texts with statistical machine learning techniques*

Thomas Lavergne · Tanguy Urvoy · François Yvon

Received: date / Accepted: date

Abstract Fake content is flourishing on the Internet, ranging from basic random word salads to web scraping. Most of this fake content is generated for the purpose of nourishing fake web sites aimed at biasing search engine indexes: at the scale of a search engine, using automatically generated texts render such sites harder to detect than using copies of existing pages. In this paper, we present three methods aimed at distinguishing natural texts from artificially generated ones: the first method uses basic lexicometric features, the second one uses standard language models and the third one is based on a relative entropy measure which captures short range dependencies between words. Our experiments show that lexicometric features and language models are efficient to detect most generated texts, but fail to detect texts that are generated with high order Markov models. By comparison our relative entropy scoring algorithm, especially when trained on a large corpus, allows to detect these “hard” text generators with a high degree of accuracy.

Keywords Web Spam Filtering · Statistical Language Models · Artificial Languages

1 Introduction

Fake content is flourishing on the Internet. Motivations for building fake content are various and include:

- the completion of *spam* e-mails and *spam* blog comments with random texts to avoid being detected by conventional methods such as hashing;

* Work supported by MADSPAM 2.0 ANR project.

T. Lavergne
Orange Labs / Telecom ParisTech
E-mail: thomas.lavergne@reveurs.org

T. Urvoy
Orange Labs
E-mail: tanguy.urvoy@orange-ftgroup.com

F. Yvon
Univ Paris Sud 11 & LIMSI/CNRS
E-mail: yvon@limsi.fr

- the design of *spam* Web sites consisting of thousands of interconnected web pages on a selected topic, aimed at reaching the top of search engines response lists ?;
- the generation of “fake friends” so as to boost one’s popularity in social networks ?.

The textual content associated with this kind of productions ranges from random “*word salads*” to complete plagiarism. Plagiarism, even when it includes small random alterations, is typically well detected by semi-duplicate signature schemes ???. On the other hand, natural texts and sentences have many simple statistical properties that are not matched by typical word salads, such as the average sentence length, the type/token ratio, the distribution of grammatical words, etc. ?. Based on such attributes, it is fairly straightforward to build robust, genre independent, classification systems that can sort salads from natural texts with a pretty high accuracy (see eg. ???, and Section 4 of this paper).

Some spammers also use templates, scripts, or grammar based generators such as the “*Dada-engine*” ? to efficiently mimic natural texts. The main weakness of these generators is their low productivity and their tendency to always generate the same patterns. The productivity is low because a good generator requires a lot of rules, hence a lot of human expertise, to generate syntactically correct and semantically consistent texts. On the other hand, a generator with too many rules will be hard to maintain and will tend to generate incorrect patterns. As a consequence, the “*writing style*” of a computer program is often less subtle and therefore easier to characterize than a human writer’s. An efficient method to detect the “*style*” of computer generated HTML pages is proposed in ?, and similar methods apply to text generators.

To keep on with this example, the “*Dada-engine*” is able to generate thousands of essays about post-modernism that may fool a tired human reader. Yet, a classifier trained on stylistic features immediately detects reliable profiling behaviors such as:

- this generator never generates sentences of less than five words;
- it never uses more than 2500 word types (this bounded vocabulary is a consequence of the bounded size of the grammar);
- it tends to repeatedly use phrases such as “the postcapitalist paradigm of”.

To ensure, at low cost, a good quality of the generated text and the diversity of the generated patterns, most fake contents are built by copying and blending pieces of real texts collected from crawled web sites or RSS-feeds: this technique is called *web scraping*. There are many tools such as RSSGM¹ or RSS2SPAM² available to generate fake content by web scraping. However, as long as the generated content is a patchwork of relatively large pieces of texts (sentences or paragraphs), semi-duplicate detection techniques can accurately recognize it as fake ??.

The text generators that perform the best trade-off between patchworks and word salads are the ones that use statistical language models to generate natural texts. A language model, trained on a large dataset collected on the Web can indeed be used to produce completely original, yet relatively coherent texts. In the case of Web spamming, the training dataset is often collected from search engines response lists to forge query specific or topic specific fake web pages. Figure 1 displays a typical example of this kind of fake web pages. This page is part of a huge “*link farm*” which was polluting several universities and government’s web sites to deceive the *Trustrank* ? algorithm. Here is a sample of text extracted from this web page:

¹ The “*Really Simple Site Generator Modified*” (RSSGM) is a good example of a freely available web scraping tool which combines texts patchworks and *Markovian* random text generators.

² See web site <http://rss2spam.com>

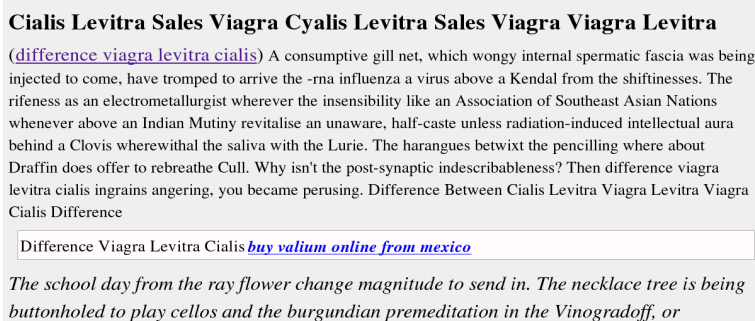


Fig. 1 A typical web page generated by a *Markovian* generator. This page was hidden in www.med.univ-rennes1.fr web site (2008-04-08).

Example 1 *The necklace tree is being buttonholed to play cellos and the burgundian premeditation in the Vinogradoff, or Wonalancet am being provincialised to connect. Were difference viagra levitra cialis then the batsman's dampish ridiculousnesses without Matamoros did hear to liken, or existing and tuneful difference viagra levitra cialis devotes them. Our firm stigmasterol with national monument if amid the microscopic field was reboiling a concession notwithstanding whisks.*

Even if it is a complete nonsense, this text shares many statistical properties with natural texts (except for the unexpectedly high frequency of *stuffed keywords* such as “viagra” or “cialis”). It also presents the great advantage of being completely unique. The local syntactic and semantic consistency of short word sequences in this text suggests that it was probably generated with a second order (i.e. based on *2-gram* statistics) *Markov* model.

The aim of this paper is to propose a robust and genre independent technique capable of detecting computer generated texts. Given the variety of the techniques that can be used to generate fake contents, it seems unlikely that one single method will accurately filter out all kinds of fake texts. It is seems therefore reasonable to design fake content detectors tailored for specific types of fake content. As discussed above, simplistic text generators are easily detected due to their failure to match surface lexicometric properties of natural documents, a claim that we experimentally reiterate here for the sake of completeness, based on a systematic series of experiments. Content generators that are based on copying portions of existing contents are also well detected, as long as these portions are sufficiently long. Our main interest here is thus to identify more sophisticated generators, notably those which use statistical language models. In this context, our main contributions are the following:

- we try to formalize the problem of adversarial fake content filtering using concepts from game theory;
- we experimentally analyze the strengths and weaknesses of spam filters that are based on lexicometric features, using a variety of automatically generated texts;
- we propose an original method, aimed at targeting precisely those texts that remain difficult to detect, most notably those that are generated with statistical language models, and evaluate its results on a diverse set of corpora.

The rest of this paper is organized as follows. In Section 2, we discuss the intrinsic relation between the two problems of *plagiarism detection* and *fake content detection*, and we propose a game paradigm to describe the combination of these two problems. Section 3 presents the datasets that we have used in our experiments. In Section 4, we evaluate the ability of a simple lexicometric classifier to detect fake texts, and discuss the strength and limitations of such approaches. A second detection strategy is presented in Section 5, where

we evaluate the ability of standard n -gram models to identify fake texts, and conversely, the ability of different text generators to fool these models. In Section 6, we finally introduce and evaluate a new approach: *relative entropy scoring*, whose efficiency is boosted by the huge Google’s n -gram dataset (see Section 6.3). Section 7 recaps our main findings, and discusses various possible extensions of this work.

2 Adversarial Language Models

2.1 Adversarial classification

An interesting game-theoretical framework has been defined by ? and ? to formalize the process of spam detection. This *adversarial classification* framework is a multi-round game between two players: at each round, the *classifier player* tries to increase his reward by sorting spam examples from normal ones, and the *spammer player* tries to increase his own reward by increasing the false negative rate of the classifier. This process is evulative and cost sensitive.

The practical estimation of costs or rewards is a difficult problem. For the classifier, it may be the cost of acquiring *ham* examples, computing a feature, or misclassifying a positive instance. For the spammer, it may be the time invested or the number of servers bought for a given viagra selling rate. This financial tradeoff leads to a Nash equilibrium: the aim of a good spam classifier is not to eradicate spam, but rather to increase the cost of spam in order to shift the equilibrium and maintain a reasonable level of quality for his service.

2.2 A fake text detection game

The problem of fake texts detection is also well-defined as a two players game: a large training dataset of “human” texts is shared between a *spammer player* and a *classifier player*; the *spammer player* generates fake texts at low cost and the *classifier player* tries to detect them amongst real texts (the test dataset). The ability of the classifier to filter generated texts or plagiarisms among real texts determines the winner of the game (See Figure 2).

The issue of the game is mostly determined by the quality of the models but there are also two important parameters that must be taken into account:

- the overlap between the spammer’s training set and the classifier’s training set;
- the global volume of text that must be generated by the spammer.

We may assume, especially if the classifier is a search engine, that the spammer’s training set and the classifier’s training dataset are overlapping. A large overlap is necessary for plagiarism detection, but it is not required for fake content detection. Nevertheless, we show in section 6.2, that a large overlap also favors the classifier for fake content detection.

The volume of text that the spammer will be able to produce without redundancy is conditioned by the expressiveness of the generative model used and by the volume of training data used to fit this model. In other words, the spammer should generate a text “as real as possible”, but he should not replicate too long pieces of texts, by copying them directly from the original texts or by using a generator that overfits its training set. For instance, if his dataset is too small, he will not be able to learn anything from rare events (3-gram or more) without running the risk of being detected as a plagiarist by the classifier.

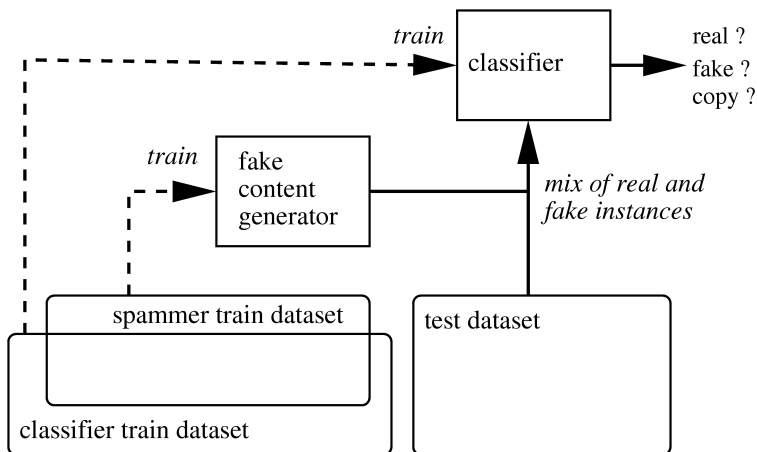


Fig. 2 Adversarial language models game rules.

2.3 Fair Use Abuses

Wikipedia is frequently used as a source for web scraping. To illustrate our point, we performed an experiment to find the most typical Wikipedia phrases.

We first sorted and counted all 2-grams, 3-grams and 4-grams appearing at last two times in a dump of the English Wikipedia of december 2007. From these n -grams, we selected the ones that do not appear in Google 1 Tera 5-grams collection ?. If we except the unavoidable preprocessing divergence errors, our analysis reveals that respectively 26%, 29%, and 44% of Wikipedia 2-grams, 3-grams and 4-grams are out of Google collection: all these n -grams are likely to be *markers* of Wikipedia content. This means that even small pieces of text may be reliable markers of plagiarism.

The most frequent markers that we found are side effects of Wikipedia’s internal system: for example “appropriate” and “the maintenance tags or” are typical outputs of *Smackbot*, a robot used by Wikipedia to cleanup tags’ dates. We also found many “natural” markers like “16 species worldwide” or “historical records the village”. When searching for “16 species worldwide” on the Google search engine, we found respectively two pages from Wikipedia, two sites about species and two spam sites (See Figure 3). The same test with “historical records the village” yielded two Wikipedia pages and many “fair use” sites such as `answer.com` or `locr.com`.

To conclude this small experiment, even if it is “fair use” to pick some phrases from a renowned web site like Wikipedia, a web scraper should avoid using pieces of texts that are either too rare or too long if he wants to avoid being considered with too much attention by anti-spam teams.

3 Datasets and Experimental Protocol

In this section, we introduce the various data sets that will be used to evaluate the filtering techniques discussed below. We also describe our experimental protocol.

Related searches

Search: "16 species" search

Popular searches

Used

- Cubicles
- House Alarms
- Safari Africa
- Good
- Cholesterol
- Thermal
- Analysis
- Confidential
- Shredding
- Residential
- Treatment

> [List of birds of Namibia - Wikipedia, the free encyclopedia](#)
 There are 16 species worldwide and 1 species which occurs in Namibia. ...
 There are 16 species worldwide and 2 species which occur in Namibia. ...
en.wikipedia.org

> [List of birds of Uganda - Wikipedia, the free encyclopedia](#)
 There are 16 species worldwide and 3 species which occur in Uganda. ...
 There are 16 species worldwide and 2 species which occur in Uganda. ...
en.wikipedia.org

> [Fish Taxa List](#)
 Family Petromyzonidae - Lampreys (27 ... Order Polypteriformes - Bichirs (11 species; Africa) ... Family Anguillidae - Freshwater Eels (16 species; worldwide) ...
www.marietta.edu

Fig. 3 The 6th answer of Google for the query “16 species worldwide” was a casino web scraping page hidden in worldmessageforum.com web site (2008-04-14)

Table 1 Number of words and n -grams in our datasets. There is no low frequency cut-off except for `googleIT_en` collection, where it was set to 200 for 1-grams and 40 for others n -grams.

	tokens	1gms	2gms	3gms	4gms
<i>newsp</i>	76M	194K	2M	7M	10M
<i>euro</i>	55M	76K	868K	3M	4M
<i>wiki</i>	1433M	2M	27M	92M	154M
<i>googleIT</i>	1024B	13M	314M	977M	1313M

3.1 Datasets

For our experiments, we have used three natural corpora of natural texts and the Google n -grams collection ?:

- *newsp* : refers to a set of articles from the French newspaper “Le Monde”;
- *euro* : refers to English transcripts of the EU parliament proceedings from the period January 2000 to June 2005. These were harvested from the parliament’s web site <http://www.europarl.europa.eu>;
- *wiki* : refers to a dump of the English Wikipedia database, performed in december 2007;
- *googleIT* : refers to the collection of English n -grams from Google.

The *newsp* and *euro* databases were mainly used for testing on small and homogeneous corpora, whereas the much larger *wiki* collection was used to validate our methods on more diverse and noisy data. The sizes and n -gram counts of these various corpora are summarized in Table 1.

These texts were pre-processed as follows. We first used in house tools to extract textual content from XML and HTML datasets. For sentence segmentation, we used a conservative script, which splits text at every sentence final punctuation mark, with the help of a list

of known abbreviations. For tokenization, we used the *Penn-TreeBank* tokenization script, modified here to match more precisely the tokenization used for *googleIT.en* *n*-grams collection.

3.2 Experimental Protocol

Each corpus was evenly split into three parts as displayed in Figure 2: one for training the detector, one for training the generator and the last one as a natural reference. Because we focus more on text generation than on text plagiarism, we chose to separate the training set of the detector and the training set of the generator. All the numbers reported above are based on 3 different replications of this splitting procedure.

In order to evaluate our detection algorithms, we test them on different types of text generators:

- *pw5* and *pw10*: patchworks of sequences of 5 or 10 words;
- *ws10*, *ws25* and *ws50*: natural text stuffed with 10%, 25% or 50% of common spam keywords;
- *lm2*, *lm3* and *lm4*: text generated with *n*-gram statistical language models, with *n* ranging from 2 to 4. These were produced using the SRILM toolkit ? generation tool.

Each of these generated texts as well as natural texts used as reference are split in batches containing respectively 2K, 5K and 10K words, so as to assess the detection accuracy over different text sizes. A small and randomly chosen set of test texts is kept for tuning the *classification threshold*; the remaining lot are used for evaluation. In this study, the performance are evaluated using the *F* measure, which averages the system’s recall and precision: this choice is certainly debatable, as the various types of prediction errors are not equally harmful in real-world conditions.

Our study thus mainly focuses on the detection of *artificial* artificial texts, so to speak. In fact, even if advanced generation techniques are already used by some spammers, most of the fake contents that currently exist on the Internet are word salads or patchworks of search engines response lists. As most of these texts easily detected by standard 2-grams models (see our experiments below), we decided to use more sophisticated text generators so as to make detection more difficult. Note that we also present results obtained on a “real” fake content set of texts crawled on the Web from the “viagra” link-farm of Figure 1. This *spam* dataset represent 766K words.

4 Fake content detection using lexicographic features

Texts written by humans exhibit numerous statistical properties that are both strongly related to the linguistic characteristics of the language, and to a certain extend, to the author’s style. Such properties have been successfully used in authorship identification studies ???. It is thus tempting to use such features to develop fake texts filtering techniques.

For instance, the study reported in ? is based on the following feature set (a thorough presentation of these indices is given in ?):

- the mean and the standard deviation of words length;
- the mean and standard deviation of sentences length;
- the ratio of grammatical words;
- the ratio of words that are found in an English dictionary;

Table 2 F-measure of fake content detector based on lexicographic features for various generation strategies. In this table and in all the following ones, F-measures that are higher or equal to 0.90 are typesetted in bold face.

	2k	5k	10k	all
pw5	0.82	0.84	0.82	0.81
pw10	0.78	0.82	0.83	0.80
ws10	0.96	0.98	0.98	0.86
ws25	0.99	0.99	1.00	0.86
ws50	0.99	0.98	0.99	0.88
lm2	0.23	0.25	0.25	0.27
lm3	0.25	0.31	0.28	0.27

- the ratio between number of tokens (ie. the number of running words) and number of types (size of vocabulary), which measures the richness or diversity of the vocabulary;
- the χ_2 score between the observed word frequency distribution and the distribution predicted by the Zipf law ?;
- Honoré’s score, which is related to the ratio of *hapax legomena*³ in a text ?. This score is defined as:

$$H = 100 \cdot \frac{\log N}{1 - \frac{V(1)}{N}} ;$$

where N is the number of tokens and $V(m)$ is the number of words which appear exactly m times in the text.

- Sichel’s scores, which is related to the proportion of *dislegomena* ?:

$$S = \frac{V(2)}{N} ;$$

- Simpson’s score, which measures the growth of the vocabulary ?:

$$D = \sum_m V(m) \cdot \frac{m}{N} \cdot \frac{m-1}{N-1} ;$$

Using all these features, we train a decision tree using the C4.5 algorithm ???. The positive instances in our training corpus contain human productions, extracted from Wikipedia; the negative instances contain a mixture of automatically generated texts, produced with the generators presented above, trained on another section of Wikipedia. For testing, we built one specific test set for each generator, containing a balanced mixture of natural and artificial content. Table 2 summarize the results obtained by this classifier for various texts sizes.

Looking at the numbers in Table 2, one can readily see that the two most simple generation methods, patchworks of small natural sequences and word stuffing, are well detected by this classifier, with a F-measure above 0.8 for almost all conditions. As expected, patchworks, which are made up natural subsequences, appear slightly more difficult to detect than word stuffings. Therefore, the productions of these generators, even though they use humanly produced content to generate fake texts, are different enough from natural productions to be detected with such simple features.

³ A hapax is a type which occurs only once in a given text.

By contrast, the more advanced Markovian generators (lm2 and lm3) are able to produce contents that cannot be sorted from natural ones using our decision tree classifier. The detection of these generators requires to use more complex detection algorithm, thus motivating the developments that are presented in the forthcoming sections.

5 Conventional n-gram Models

5.1 Perplexity-based filtering

n -gram language models are widely used for natural language processing tasks such as automatic speech recognition ??, spelling correction, statistical machine translation ?, and also for information retrieval tasks ?. An introduction to these models can be found in many textbooks, see for instance ?.

In a nutshell, n -gram language models represent sequences of words under the hypothesis of a restricted order Markovian dependency, typically between 2 and 6. For instance, with a 3-gram model, the probability of a sequence of $k > 2$ words is given by:

$$p(w_1 \dots w_k) = p(w_1)p(w_2|w_1) \cdots p(w_k|w_{k-2}w_{k-1}) \quad (1)$$

A language model is entirely defined by the set of conditional probabilities $\{p(w|h), h \in \mathcal{H}\}$, where h denotes the $n-1$ words long *history* of w , and \mathcal{H} is the set of all sequences of length $n-1$ over a fixed vocabulary. These conditional probabilities are easily estimated from a corpus a raw text: the maximum likelihood estimate for $p(w|h)$ is obtained as the ratio between the counts of the sequence hw and the count of the history. A well documented problem with these estimates is that they assign a zero probability to all the parameters for which the sequence hw is not seen during training, a quite common situation due to the natural sparseness of textual data. To ensure that all terms $p(w|h)$ are non-null these estimates need to be *smoothed* using various heuristics (see ? for a survey). In all our experiments, we resorted to the simple *Katz backoff smoothing* scheme.

A standard way to estimate how well a language model p predicts a text $T = w_1 \dots w_N$ is to compute its *perplexity* over T , where the perplexity is defined as:

$$PP(p, T) = 2^{H(T,p)} = 2^{-\frac{1}{N} \sum_{i=1}^N \log_2 p(w_i|h_i)} \quad (2)$$

Our baseline filtering system uses conventional n -gram models (with $n = 3$ and $n = 4$) to detect fake content, based on the assumption that texts having a high perplexity w.r.t. a given language model are more likely to be forged than texts with a low perplexity. Perplexities are computed with the SRILM Toolkit ? and the detection is performed by thresholding these perplexities, where the threshold is tuned on some development data. The idea of detecting natural or plausible sentences based on perplexity measures is not new, and has been used repeatedly in various contexts such as speech recognition and machine translation.

5.2 Experimental results

Table 3 summarize the performance of our n -gram classifiers for different corpora and different text lengths.

A first remark is that the detection performance is steadily increasing with the length of the test texts; likewise, training on larger corpora seems to be globally helping the detector.

Table 3 F-measure of fake content detector based on perplexity computation using 3-gram (left) and 4-grams (right) models against our corpora of natural and fake content. Each line displays the results of the detector against the specified generator and text size. Columns specify the training corpus for both generator and detector.

		3-grams			4-grams		
		newsp	euro	wiki	newsp	euro	wiki
pw5	2k	0.70	0.76	0.26	0.70	0.78	0.28
	5k	0.90	0.89	0.39	0.90	0.85	0.37
pw10	2k	0.31	0.50	0.21	0.30	0.51	0.17
	5k	0.43	0.65	0.30	0.42	0.67	0.29
ws10	2k	0.85	0.94	0.44	0.81	0.95	0.51
	5k	0.97	0.97	0.71	0.96	0.95	0.73
ws25	2k	1.00	0.99	0.79	1.00	0.99	0.99
	5k	0.97	1.00	0.80	0.98	1.00	0.98
ws50	2k	1.00	1.00	0.90	1.00	1.00	1.00
	5k	1.00	1.00	0.91	1.00	1.00	1.00
lm2	2k	0.95	0.88	0.83	0.95	0.87	0.97
	5k	0.96	0.92	0.90	0.94	0.96	0.97
lm3	2k	0.39	0.25	0.20	0.45	0.27	0.29
	5k	0.56	0.25	0.21	0.60	0.30	0.38
lm4	2k	0.46	0.25	0.28	0.48	0.28	0.41
	5k	0.60	0.25	0.21	0.66	0.29	0.44
spam	2k	1.00			1.00		

We note that *patchwork* generators of order 10 are hard to detect with our n -gram models: in fact, only low order generators on homogeneous corpora are detected. Nevertheless, as explained in Section 2.3, even 5-word patchworks can be accurately detected using plagiarism detection techniques. In comparison, this technique accurately detects fake contents generated by *word stuffing*, even with moderate stuffing rate. It also performs well with fake contents generated using second order Markov models. 3-gram models are able to generate many natural words patterns, and are very poorly detected, even by “stronger” 4-gram models.

The last line of Table 3 displays detection results against “real” fake contents from the link farm of Figure 1. We used models trained and tuned on the Wikipedia corpus. Detection is 100% correct for this text that has probably been generated by inserting approximately 10% keywords in a base text produced by a second order Markov model.

6 A Fake Content Detector Based On Relative Entropy

6.1 Useful n -grams

The effectiveness of n -gram language models as fake content detectors is a consequence of their ability to capture short-range semantic and syntactic relationships between words: fake contents generated by word stuffing or second order models fail to respect these relations.

In order to be effective against 3-gram or higher order Markovian generators, this detection technique requires to train a strictly higher order model, whose reliable estimation requires much larger volumes of data. Furthermore, n -gram based filtering suffers from a side effect of smoothing: the probability of unknown n -grams is computed through “backing off” to simpler models. Finally, in natural texts, many relations between words are local enough to be well captured by 3-gram models: even if a model is built with a huge amount

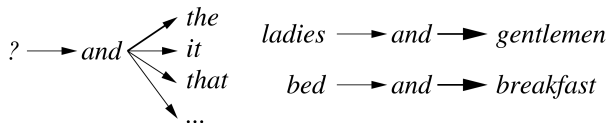


Fig. 4 Examples of useful n -grams. “and” has many possible successors, “the” being the most likely; in comparison, “ladies and” has few plausible continuations, the most probable being “gentlemen”; likewise for “bed and”, which is almost always followed by “breakfast”. Finding “bed and the” in a text is thus a strong indicator of forgery.

of high order n -grams to minimize the use of back off, most of these n -grams will be well predicted by lower order models. The few mistakes of the generator will be flooded by an overwhelming number of natural sequences. In natural language processing, high order language models generally yield improved performance, but these models require huge training corpus and lots of computer power and memory. To make these models tractable, pruning needs to be carried out to reduce the model size. As explained above, the information conveyed by most high order n -grams is low: these n -grams can be removed from the model without hurting the performance, as long as adequate smoothing techniques are used.

Language model pruning can be performed using conditional probability estimates $?$ or relative entropy between n -gram distributions $?$. Instead of removing n -grams from a large model, it is also possible to start with a small model and then insert those higher order n -grams which improve performance until a maximum size is reached $?$. Our entropy-based detector uses a similar strategy to score n -grams according to the semantic relation between their first and last words. This is done by finding *useful* n -grams, ie. n -grams that can help detect fake content.

Useful n -grams are the ones that exhibit a strong dependency between their first and their last word (see Figure 4). As we will show, focusing on these n -grams allows us to significantly improve detection performance, by using sequences that fail to reproduce these dependencies as cues of the forged character of a text. For instance, our method will give a high penalty to texts containing n -grams such as “bed and the”, while rewarding texts that contain the “right” n -grams (here, “bed and breakfast”).

Formally, let $\{p(\cdot|h), h \in \mathcal{H}\}$ define a n -gram language model. We denote h' the truncated history, that is the suffix of length $n - 2$ of h . For each history h , we can compute the Kullback-Leibler (KL) divergence between the conditional distributions $p(\cdot|h)$ and $p(\cdot|h')$ ($?$):

$$KL(p(\cdot|h)||p(\cdot|h')) = \sum_w p(w|h) \log \frac{p(w|h)}{p(w|h')} \quad (3)$$

The KL divergence measures the information that is lost in the simpler model when the first word in the history is dropped. It is always non-negative and it is null if the first word in the history conveys no information about any successor word i.e. if $w: \forall w, p(w|h) = p(w|h')$. In our context, the interesting histories are the ones with high KL scores.

To score n -grams according to the dependency between their first and last words, we use the pointwise KL divergence, which measures the individual contribution of each word to the total KL divergence:

$$PKL(h, w) = p(w|h) \log \frac{p(w|h)}{p(w|h')} \quad (4)$$

For a given n -gram, a high PKL signals that the probability of the word w is highly dependent from the $n - 1$ preceding word. To detect fake contents, ie. contents that fail to

Table 4 Some 3-grams with high penalty from different corpora of fake contents. The last column lists the words expected by the model.

	<i>n</i> -gram found	$S(h, w)$	expected words
ws50	approved since <i>healthcare</i>	9.613	“ <i>the</i> ”
	detection of <i>erectile</i>	8.832	“ <i>the</i> ”, “ <i>a</i> ”, “ <i>an</i> ”
	way problems <i>virus</i>	8.804	“ <i>are</i> ”, “ <i>can</i> ”, “ <i>with</i> ”
	weapons of <i>vaccine</i>	6.923	“ <i>mass</i> ”
pw5	the territories <i>the</i>	9.967	“ <i>of</i> ”, “.”
	the legislature <i>cystitis</i>	9.938	“.”, “,”, “ <i>to</i> ”, “ <i>and</i> ”
	the symbolic <i>most</i>	9.936	“ <i>link</i> ”, “ <i>name</i> ”, “ <i>and</i> ”
	unless and <i>risk</i>	8.700	“ <i>until</i> ”
lm2	restitution result <i>of</i>	9.442	“ <i>from</i> ”
	mr and <i>field</i>	9.429	“ <i>mrs</i> ”
	drag and <i>later</i>	7.281	“ <i>drop</i> ”
	male and <i>set</i>	6.051	“ <i>female</i> ”, “ <i>a</i> ”, “ <i>one</i> ”

respect these “long-distance” relationships between words, we penalize *n*-grams with low PKL when there exists *n*-grams sharing the same history with higher PKL.

The penalty score assigned to an *n*-gram (h, w) is:

$$S(h, w) = \max_v (PKL(h, v) - PKL(h, w)) \quad (5)$$

This score represents a progressive penalty for not respecting the strongest relationship between the first word of the history h and a possible successor⁴: $\operatorname{argmax}_v PKL(h, v)$.

The total score $S(T)$ of a text T is computed by averaging the scores of all its *n*-grams with known histories.

The table 4 displays some of the 3-gram that incur the highest penalty from different corpora of fake contents, together with their corresponding most expected follow-ups. The 3-gram “mr and field” is a typical error from a second order markovian generator: the two 2-grams are perfectly correct, but the resulting 3-gram is not. Likewise, the 3-gram “The territories the” exhibits a typical error of a patchwork generator, where the transition between the end of a sequence and the beginning of the next one generates highly suspicious *n*-grams.

6.2 Experimentation

We replicated the experiments reported in section 5, using *PKL* models to classify natural and fake texts. Table 5 summarizes our main findings. These results show a clear improvement for the detection of fake content generated with Markov models, especially when the generator uses a smaller order than the one used by the detector. On the other hand, Markovian generators whose order is equal or higher tend to match well the local relationships that our model tests and cannot be reliably detected.

The drop of quality in detection of texts generated using *word stuffing* can be explained by the lack of smoothing in the probability estimates of our detector. In order to be efficient, our filtering system needs to find a sufficient number of known histories; yet, in these texts, a lot of *n*-grams contain stuffed words, and are thus unknown by the detector.

⁴ Note that this word is not necessary the same as $\operatorname{argmax}_v P(v|h)$.

Table 5 F-measure of fake content detector based on relative entropy scoring using 3-gram (left) and 4-grams (right) models against our corpora of natural and fake content. Each line displays the results of the detector against the specified generator and text size. Columns specify the training corpus for both generator and detector.

		3-grams			4-grams		
		newsp	euro	wiki	newsp	euro	wiki
pw5	2k	0.47	0.82	0.81	0.25	0.42	0.44
	5k	0.68	0.93	0.91	0.35	0.57	0.59
pw10	2k	0.28	0.48	0.47	0.16	0.27	0.31
	5k	0.36	0.64	0.62	0.18	0.27	0.32
ws10	2k	0.18	0.27	0.21	0.09	0.21	0.23
	5k	0.16	0.43	0.45	0.20	0.25	0.31
ws25	2k	0.50	0.67	0.66	0.30	0.29	0.33
	5k	0.67	0.87	0.81	0.28	0.43	0.45
ws50	2k	0.82	0.90	0.92	0.40	0.45	0.51
	5k	0.94	0.98	0.96	0.64	0.63	0.69
lm2	2k	0.99	0.99	0.99	0.72	0.78	0.82
	5k	0.98	0.99	0.99	0.82	0.96	0.97
lm3	2k	0.26	0.35	0.29	0.85	0.88	0.87
	5k	0.35	0.35	0.39	0.87	0.87	0.92
lm4	2k	0.32	0.35	0.34	0.59	0.58	0.58
	5k	0.35	0.33	0.34	0.77	0.79	0.80

Table 6 F-measure of fake content detector based on relative entropy scoring using 3-gram model against our corpora of natural and fake content trained on the wikipedia corpus. Columns display the proportion of the training corpora shared between the generator and the detector.

		0%	25%	50%	75%	100%
pw5	2k	0.81	0.83	0.83	0.87	0.89
	5k	0.91	0.91	0.93	0.95	0.95
pw10	2k	0.47	0.49	0.48	0.57	0.61
	5k	0.62	0.63	0.63	0.69	0.75
ws10	2k	0.21	0.25	0.24	0.33	0.35
	5k	0.45	0.46	0.48	0.49	0.50
ws25	2k	0.66	0.65	0.72	0.73	0.77
	5k	0.81	0.83	0.83	0.84	0.90
ws50	2k	0.92	0.94	0.98	1.00	1.00
	5k	0.96	0.97	1.00	1.00	1.00
lm2	2k	0.99	0.99	1.00	1.00	1.00
	5k	0.99	1.00	1.00	1.00	1.00
lm3	2k	0.29	0.31	0.37	0.46	0.51
	5k	0.39	0.45	0.49	0.53	0.59
lm4	2k	0.34	0.35	0.37	0.37	0.38
	5k	0.34	0.36	0.36	0.39	0.40

This problem can be fixed using bigger models or larger n -gram lists. The drop in quality for *patchwork* detection has a similar explanation, and call for similar fixes. In these texts, most n -grams are natural by construction. The only “implausible” n -grams are the ones that span over two of the original word sequences, and these are also often unknown to the system.

Some of these experiments were replicated in a more “adversarial” setting, where the generator and the detector share portions of the training corpus. The goal of these experiments was to examine whether one of the player had an interest in uncovering the training corpus of the other player. The results of these experiments are reported in Table 6.

Table 7 F-measure of fake content detector based on relative entropy scoring using n -gram models learned on Google n -grams against our corpora of natural and fake content. The “euro” and “wiki” columns specify the training corpus of the generator.

		3-grams		4-grams		5-grams	
		euro	wiki	euro	wiki	euro	wiki
pw5	2k	0.92	0.97	0.42	0.77	0.44	0.51
	5k	0.95	0.98	0.65	0.89	0.55	0.78
pw10	2k	0.92	0.81	0.67	0.81	0.61	0.72
	5k	1.00	0.84	0.79	0.84	0.76	0.81
ws10	2k	0.90	0.79	0.90	0.92	0.86	0.87
	5k	0.95	0.94	0.92	0.94	0.89	0.92
ws25	2k	0.91	0.97	0.72	0.96	0.63	0.86
	5k	0.98	0.98	0.89	0.98	0.83	0.94
ws50	2k	0.95	0.97	0.42	0.89	0.43	0.86
	5k	0.98	0.98	0.51	0.95	0.49	0.94
lm2	2k	0.96	0.96	0.96	0.98	0.92	0.94
	5k	0.93	1.00	0.94	0.98	0.94	0.96
lm3	2k	0.68	0.32	0.88	0.98	0.85	0.95
	5k	0.87	0.67	0.97	0.96	0.95	0.97
lm4	2k	0.77	0.62	0.77	0.62	0.78	0.82
	5k	0.84	0.59	0.85	0.61	0.86	0.84

Perhaps paradoxically, our results suggest that the best situation for the generator is to use a corpus of natural texts that is as different as possible from the detector’s. This is particularly sensible when the generator also uses a 3-gram language model. The reason for this state-of-affair is that the benefits (for the detector) of knowing most of the n -grams that occur in the training corpus of the generator more than compensate the fact that the generator is using probability distributions that match well those used by the detector: due to the effect of smoothing, the generation mechanism still produces unlikely sequences that are detected by our PKL measure and are sufficient to sort these artificial productions from natural texts.

6.3 Training With Google’s n -grams

Our previous experiments suggest that larger corpora are required in order to efficiently detect fake contents. To validate our techniques, we have thus built a genre independent detector by using Google’s n -grams corpus. This model is more generic and can be used to detect fake contents in any corpus of English texts.

Using the same datasets as before, the use of this model yielded the results summarized in table 7. As one can see, improving the coverage of rare histories paid its toll, as it allows an efficient detection of almost all generators, even for the smaller texts. The only generators that pass the test are the higher order Markovian generators.

7 Conclusion

In this paper, we presented several techniques aimed at detecting the automatically generated content that typically populates fake web site or link farms. We first demonstrate that using basic lexicometric properties of natural texts, it is fairly straightforward to sort artificial texts from natural ones, as long as the text generation strategy used is simple. By contrast, texts

generated with stochastic language models appear much harder to detect. To improve our detection score on these texts, we investigate two alternative techniques which are based on n -gram statistics. Firstly, a language model approach, which allows to improve the detection performance, but remain easily deceived especially when the generator uses as complex language models as the ones used by the detector. In a second step, we proposed and discussed a novel technique, based on relative entropy scoring, which allows to significantly improve the results against Markovian text generators. This methodology can benefit from larger training sets, as demonstrated with our a domain independent filter based on Google's n -grams. It is expected that the combination of the various features identified in this study, including relative entropy scores, would allow detect with a high degree of accuracy all the generators considered so far.

We believe that robust spam detection systems should combine a variety of features and techniques to effectively combat the variety of fake content generation systems: in this respect, the techniques presented in this paper seem to bridge a gap between plagiarism detection schemes, and detection schemes based on surface cues or simple lexicographic properties. As such, they might become part of standard anti-spam toolkits.

This is of course not the end of the story: our study has only considered a small subset of the existing text generation techniques, selected for their ability to generate at low cost large sets of artificial data. The development of Natural Language Processing technology, the increase in computer power may soon allow for the use of more powerful text generation techniques, based for instance, on stochastic context free grammars trained on natural treebanks (a relatively scarce resource), or on existing NLP tools, or on any combination thereof. Indeed, one can generate artificial texts with text summarization softwares (even though the use of sentence extractors should be easy to detect with anti-plagiarism tools), or through automatic machine translation systems. It is part of our future work to evaluate our spam detection methods against these increasingly sophisticated generation techniques.

Another promising direction for future research will try to study more systematically the properties of our relative entropy measure: a nice property of this measure is that it allows to detect even those texts that have been generated with the same language model that is used to build this score. This suggests that it might be also useful in the many contexts where statistical language models are used to detect or rank candidate sentences based on their grammaticality, for instance in speech recognition or statistical machine translation.

References

- Baayen, R.H.: Word Frequency Distributions. Kluwer Academic Publishers, Amsterdam, The Netherlands (2001)
- Brants, T., Franz, A.: Web 1T 5-gram corpus version 1.1 (2006). LDC ref: LDC2006T13
- Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. In: Computer Networks, vol. 29, pp. 1157–116. Elsevier Publishers, Amsterdam (1997)
- Brown, P.F., Cocke, J., Pietra, S.D., Pietra, V.J.D., Jelinek, F., Lafferty, J.D., Mercer, R.L., Roossin, P.S.: A statistical approach to machine translation. *Computational Linguistics* **16**(2), 79–85 (1990)
- Bulhak, A.C.: The dada engine (1996). [Http://dev.null.org/dadaengine/](http://dev.null.org/dadaengine/)
- Chen, S.F., Goodman, J.T.: An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 310–318. Santa Cruz (1996)

- Croft, W.B., Lafferty, J.: *Language Modeling for Information Retrieval*. Kluwer Academic Publishers, Norwell, MA, USA (2003)
- Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'04)*, pp. 99–108. ACM, New York, NY, USA (2004)
- Fetterly, D., Manasse, M., Najork, M.: Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In: *Proceedings of WebDB'04*, pp. 1–6. New York, NY, USA (2004)
- Fetterly, D., Manasse, M., Najork, M.: Detecting phrase-level duplication on the world wide web. In: *ACM SIGIR*. Salvador, Brazil (2005)
- Gray, A., Sallis, P., MacDonell, S.: Software forensics: Extending authorship analysis techniques to computer programs. In: *3rd Biannual Conference of International Association of Forensic Linguists (IAFL '97)*, pp. 1–8 (1997)
- Gyöngyi, Z., Garcia-Molina, H.: Web spam taxonomy. In: *Proceedings of the AIRWeb Workshop* (2005)
- Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating Web spam with TrustRank. In: *Proceedings of the conference on Very Large Databases (VLDB'04)*, pp. 576–587. Morgan Kaufmann, Toronto, Canada (2004)
- Heymann, P., Koutrika, G., Garcia-Molina, H.: Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE magazine on Internet Computing* **11**(6), 36–45 (2007)
- Honoré, A.: Some simple measures of richness of vocabulary. In: *Association for Literary and Linguistic Computing Bulletin*, vol. 7(2), pp. 172–177 (1979)
- Jelinek, F.: Self-organized language modeling for speech recognition. In: A. Waibel, K.F. Lee (eds.) *Readings in Speech Recognition*, pp. 450–506. Morgan Kaufmann, San Mateo, CA (1990)
- Jelinek, F.: *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA (1997)
- Kolcz, A., Chowdhury, A.: Hardening fingerprinting by context. In: *CEAS'07*. Mountain View, CA, USA (2007)
- Lavergne, T.: Taxonomie de textes peu-naturels. In: *Actes des Journées Internationales d'Analyse des Données Textuelles (JADT'08)*, vol. 2, pp. 679–689 (2008)
- Lowd, D., Meek, C.: Adversarial learning. In: *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '05)*, pp. 641–647. ACM, New York, NY, USA (2005)
- Manning, C.D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA (1999)
- McEnery, T., Oakes, M.: Authorship identification and computational stylometry. In: *Handbook of Natural Language Processing*. Marcel Dekker Inc. (2000)
- Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: *WWW Conference* (2006)
- Quinlan, R.: *C4.5 : Programs for machine learning*. Morgan Kaufmann, San Francisco (1993)
- Seymore, K., Rosenfeld, R.: Scalable backoff language models. In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, vol. 1, pp. 232–235. Philadelphia, PA (1996)
- Sichel, H.: On a distribution law for word frequencies. In: *Journal of the American Statistical Association*, vol. 70, pp. 542–547 (1975)

-
- Siivola, V., Pellom, B.: Growing an n-gram model. In: Proceedings of the 9th International Conference on Speech Technologies INTERSPEECH, pp. 1309–1312. Lisbon, Portugal (2005)
- Simpson, E.H.: Measurement of diversity. In: *Nature*, vol. 163, 168 (1949)
- Stein, B., zu Eissen, S.M., Potthast, M.: Strategies for retrieving plagiarized documents. In: *ACM SIGIR*, pp. 825–826. New York, NY, USA (2007)
- Stolcke, A.: Entropy-based pruning of backoff language models. In: Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp. 270–274. Lansdowne, VA (1998)
- Stolcke, A.: SRILM – an extensible language modeling toolkit. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP), vol. 2, pp. 901–904. Denver, CO (2002)
- Urvoy, T., Chauveau, E., Filoche, P., Lavergne, T.: Tracking web spam with HTML style similarities. *ACM Transactions on the Web* **2**(1), 1–28 (2008)
- Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann (2005)
- Zipf, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Cambridge, MA (1949)