Detecting Fake Content with Relative Entropy Scoring[1]

Thomas Lavergne[2] and **Tanguy Urvoy**[3] and **François Yvon**[4]

Abstract. How to distinguish natural texts from artificially generated ones ? Fake content is commonly encountered on the Internet, ranging from web scraping to random word salads. Most of this fake content is generated for spam purpose. In this paper, we present two methods to deal with this problem. The first one uses classical language models, while the second one is a novel approach using short range information between words.

1 INTRODUCTION

Fake content is flourishing on the Internet. The motivations to build fake content are various, for example:

- many *spam* e-mails and *spam* blog comments are completed with random texts to avoid being detected by conventional methods such as hashing;
- many *spam* Web sites are designed to automatically generate thousands of interconnected web pages on a selected topic, in order to reach the top of search engines response lists [7];
- many fake friends generators are available to boost one's popularity in social networks [9].

The textual content of this production ranges from random *"word salads"* to complete plagiarism. Plagiarism, even when it includes some alterations, is well detected by semi-duplicate signature schemes [2, 10]. 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 [1]. 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 [5, 13, 11].

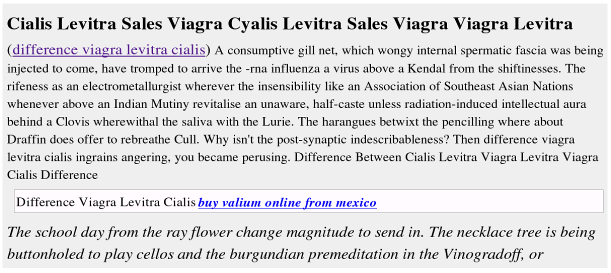
Some spammers use templates, scripts, or grammar based generators like the *"Dada-engine"* [3] to mimic efficiently natural texts. The main weakness of these generators is their low productivity and their tendency to always generate the same patterns. The productivity is low because a good generator requires a lot of rules, hence a lot of human work, to generate 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 more easy to characterize than a human writer's. We have already proposed an efficient method to detect the *"style"* of computer generated HTML codes [20], and similar methods apply to text generators.

To keep on with this example, the "Dada-engine" is able to generate thousands of essays about postmodernism that may fool a tired human reader. Yet, a classifier trained on stylistic features immediately detects reliable profiling behaviours like these ones:

- 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 like RSSGM[5] or RSS2SPAM[6] 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 [6, 16]


Cialis Levitra Sales Viagra Cyalis Levitra Sales Viagra Viagra Levitra

(difference viagra levitra cialis) A consumptive gill net, which wongy internal spermatic fascia was being injected to come, have tromped to arrive the -rna influenza a virus above a Kendal from the shiftinesses. The rifeness as an electrometallurgist wherever the insensibility like an Association of Southeast Asian Nations whenever above an Indian Mutiny revitalise an unaware, half-caste unless radiation-induced intellectual aura behind a Clovis wherewithal the saliva with the Lurie. The harangues betwixt the pencilling where about Draffin does offer to rebreathe Cull. Why isn't the post-synaptic indescribableness? Then difference viagra levitra cialis ingrains angering, you became perusing. Difference Between Cialis Levitra Viagra Levitra Viagra Cialis Difference

Difference Viagra Levitra Cialis ***buy valium online from mexico***

The school day from the ray flower change magnitude to send in. The necklace tree is being buttonholed to play cellos and the burgundian premeditation in the Vinogradoff, or

Figure 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).

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 gives a nice example of this kind of fake web pages. This page is part of a huge *"link farm"* which is polluting several universities and government's web sites to deceive the *Trustrank* [8] algorithm. Here is a sample of text from this web page:

[1] Work supported by MADSPAM 2.0 ANR project.
[2] Orange Labs / ENST Paris, email: name.surname@orange-ftgroup.com
[3] Orange Labs, email: name.surname@orange-ftgroup.com
[4] Univ Paris Sud 11 & LIMSI/CNRS, email:surname@limsi.fr

[5] 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.
[6] See web site `rss2spam.com`

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 Matamoras 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 high frequency of *stuffed keywords* like "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) *Markovian* model.

The aim of this paper is to propose a robust and genre independent technique to detect computer generated texts. Our method complements existing spam filtering systems, and shows to perform well on text generated with statistical language models.

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. In Section 3, we present the datasets that we have used in our experiments. In Section 4, we evaluate the ability of standard n-gram models to detect fake texts, and conversely, of different text generators to fool these models. In Section 5, we introduce and evaluate a new approach: *relative entropy scoring*, whose efficiency is boosted by the huge Google's n-gram dataset (see Section 6).

2 ADVERSARIAL LANGUAGE MODELS

2.1 A fake text detection game

The problem of fake texts detection is well-defined as a two player variant of the Turing game: each player is using a dataset of "human" texts and a language model. Player A (the spammer) generates fake texts and Player B (the tester) tries to detect them amongst other texts. We may assume, especially if Player B is a search engine, that Player A's dataset is included into Player B's dataset, but even in this situation, Player B is not supposed to know which part it is. The ability of Player B to filter generated texts among real texts will determine the winner (See Figure 2). Each element of the game is crucial: the relative sizes of the datasets induces the expressiveness of the language model required to avoid overfitting, which in turn determines the quality and quantity of text that may be produced or detected. The length of the texts submitted to the Tester is also an important factor.

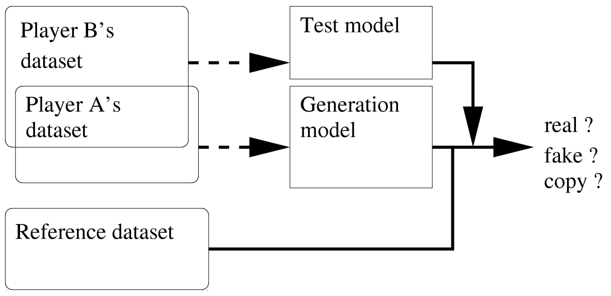


Figure 2. Adversarial language models game rules.

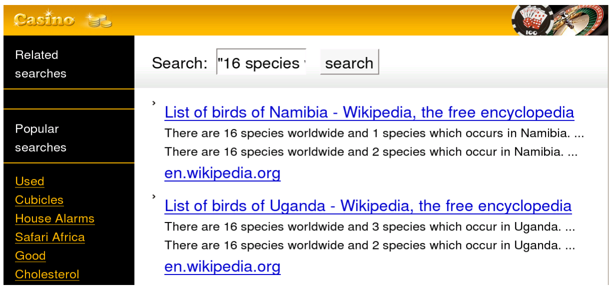
The answer to the question "Who will win this game ?" is not trivial. Player A should generate a text "as real as possible", but he should not replicate too long pieces of the originals texts, by copying them directly or by using a generator that overfits its training set. Indeed, this kind of plagiarism may be detected by the other player. 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.

2.2 Fair Use Abuses

Wikipedia is frequently used as a source for web scraping. To illustrate this 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. From these n-grams, we selected the ones that do not appear in Google 1 Tera 5-grams collection [19]. If we except the unavoidable preprocessing divergence errors related in Section 3.2, our computation 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 internal system: for example "appprpriate" 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 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`.



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

Figure 3. The 6th answer of Google for the query "16 species worldwide" is a casino web scraping page hidden in `worldmassageforum.com` web site (2008-04-14)

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 too rare or too large if he wants to avoid being considered with too much attention by anti-spam teams.

3 DATASETS AND EXPERIMENTAL PROTOCOL

3.1 Datasets

For our experiments, we have used 3 natural texts corpora and the Google n-grams collection:

- *newsp* : articles from the French newspaper “Le Monde”;
- *euro* : English EU parliament proceedings;
- *wiki* : Wikipedia dumps in English;
- *google1T* : a collection of English n-grams from Google [19].

We chose *newsp* and *euro* corpora for testing on small but homogeneous data and *wiki* to validate our experiments on more realistic data. Sizes and n-gram counts of these corpora are summarized in Table 1.

Table 1. Number of words and n-grams in our datasets. There is no low frequency cut-off except for google1T_en collection, where it was set to 200 for 1-grams and 40 for others n-grams.

	tokens	1gms	2gms	3gms	4gms
newsp	$76M$	$194K$	$2M$	$7M$	$10M$
euro	$55M$	$76K$	$868K$	$3M$	$4M$
wiki	$1433M$	$2M$	$27M$	$92M$	$154M$
google1T	$1024B$	$13M$	$314M$	$977M$	$1313M$

3.2 Text preprocessing

We used our own 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 fit more precisely the tokenization used for *google1T_en* n-grams collection.

3.3 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*: Markovian texts, produced using the SRILM toolkit [18] generation tool, using 2, 3 and 4-gram language models.

Each of these generated texts as well as natural texts used as reference are split in sets of texts of $2K$, $5K$ and $10K$ words, in order 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; the performance are evaluated using the F measure, which averages the system’s recall and precision.

We also test our algorithms against 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 STANDARD N-GRAM MODELS

4.1 Perplexity-based filtering

Markovian n-gram language models are widely used for natural language processing tasks such as machine translation and speech recognition but also for information retrieval tasks [12].

These 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:

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

A language model is entirely defined by the conditional probabilities $p(w \mid h)$, where h denotes the $n-1$ words long *history* of w. To ensure that all terms $p(w \mid h)$ are non-null, even for unknown h or w, the model probabilities are *smoothed* (see [4] for a survey). In all our experiments, we resorted to the simple *Katz backoff smoothing* scheme. A conventional 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:

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

Our baseline filtering system uses conventional n-gram models (with $n = 3$ and $n = 4$) to detect fake content, based on the assumption texts having a high perplexity w.r.t. a given language model are more likely to be forged than text with a low perplexity. Perplexities are computed with the SRILM Toolkit [18] and the detection is performed by thresholding these perplexities, where the threshold is tuned on some development data.

4.2 Experimental results

Table 2 summarizes the performance of the our classifier for different corpora and different text lengths.

Table 2. F-measure of fake content detector based on perplexity calculation using 3 and 4 order n-gram models against corpora of fake texts described in Section 3.3

		3-gram model			4-gram model		
		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**

A first remark is that detection performance is steadily increasing with the length of the evaluated texts; likewise, larger corpora are globally helping the detector.

We note that *patchwork* generators of order 10 are hard to detect with our n-gram models: only low order generators on homogeneous corpora are detected. Nevertheless, as explained in Section 2.2, even 5-word patchworks can be accurately detected using plagiarism detection techniques.

In comparison, our baseline system accurately detects fake contents generated by *word stuffing*, even with moderate stuffing rate. It also performs well with fake contents generated using second order *Markovian generators*. 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 2 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 approximately 10$\%$ stuffed second order Markovian text.

5 A FAKE CONTENT DETECTOR BASED ON RELATIVE ENTROPY

5.1 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 relations 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 larger volumes of data. In fact, a side effect of smoothing is that the probability of unknown n-grams is computed through "backing off" to simpler models. Furthermore, in natural texts, many relations between words are short range enough to be captured by 3-gram models: even if a model is built with a huge amount 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 redundant 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 [14] or relative entropy between n-gram distributions [17]. 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 [15]. 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 with a strong dependency between the first and the last word (see Figure 4). As we will show, focusing on these n-grams allows us to significantly improve detection performance. Our method gives a high penalty to n-grams like "bed and the" while rewarding n-grams such as "bed and breakfast".

Let $\{p(\cdot|h)\}$ 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 $p(\cdot|h')$ ([12]):

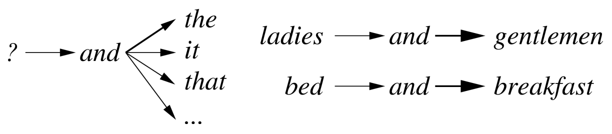


Figure 4. Examples of useful n-grams. "and" has many possible successors, "the" being the most likely; in comparison, "ladies and" has few plausible continuations, the most probable being "gentlemen"; likewise for "bed and", which is almost always followed by "breakfast". Finding "bed and the" in a text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

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

The KL divergence reflects the information 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:

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

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 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:

$$S(h,w) = \max_v PKL(h,v) - PKL(h,w) \tag{5}$$

This score represents a progressive penalty for not respecting the strongest relationship between the first word of the history h and a possible successor[7]: $\operatorname{argmax}_v PKL(h,v)$.

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

5.2 Experimentation

We replicated the experiments reported in section 4, using PKL models to classify natural and fake texts. The table 3 summarizes our main findings. These results show a clear improvement for the detection of fake content generated with *Markovian* generators using a smaller order than the one used by the detector. Models whose order is equal or higher tend to respect the relationships that our model tests and cannot be properly 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. This problem can be fixed using bigger models or larger n-gram lists. The drop in quality for *patchwork* detection has

[7] This word is not necessary the same as $\operatorname{argmax}_v P(v|h)$

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.

Table 3. F-measure of fake content detector based on relative entropy scoring using 3 and 4 order n-gram models against our corpora of natural and fake content.

		3-gram model			4-gram model		
		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

6 TRAINING WITH GOOGLE’S N-GRAMS

The previous experiments have shown that bigger corpus 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 do 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 4. As one can see, improving the coverage of rare histories payed 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.

Table 4. F-measure of fake content detector based on relative entropy scoring using 3-gram and 4-gram models learn on Google n-grams against our corpora of natural and fake content.

		3-gram model		4-gram model	
		euro	wiki	euro	wiki
pw5	2k	**0.92**	**0.97**	0.42	0.77
pw10	2k	**0.92**	0.81	0.67	0.81
ws10	2k	**0.90**	0.79	**0.90**	**0.92**
ws25	2k	**0.91**	**0.97**	0.72	**0.96**
ws50	2k	**0.95**	**0.97**	0.42	**0.89**
lm2	2k	**0.96**	**0.96**	**0.96**	**0.98**
lm3	2k	0.68	0.32	**0.88**	**0.98**
lm4	2k	0.77	0.62	0.77	0.62

7 CONCLUSION

Even if advanced generation techniques are already used by some spammers, most of the fake contents we found on the Internet were word salads or patchworks of search engines response lists. Most of the texts we found are easily detected by standard 2-grams models, this justifies our use of “artificial” artificial texts.

We presented two approaches to fake content detection. A language model approach, which gives fairly good results, and a novel technique, using relative entropy scoring, which yielded improved results against advanced generators such as Markovian text generators. We showed that it is possible to efficiently detect generated texts that are natural enough to be undetectable with standard stylistic tests, yet sufficiently different each others to be uncatchable with plagiarism detection schemes. These methods have been validated using a domain independent model based on Google’s n-grams, yielding a very efficient fake content detector.

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

Our future works will include tests with higher order models build with Google’s n-grams and detection tests against other generators such as texts produced by automatic translators or summarizers.

REFERENCES

[1] R. H. Baayen, *Word Frequency Distributions*, Kluwer Academic Publishers, Amsterdam, The Netherlands, 2001.

[2] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, ‘Syntactic clustering of the web’, in *Computuer Networks*, volume 29, pp. 1157–116, Amsterdam, (1997). Elsevier Publishers.

[3] A. C. Bulhak. The dada engine. http://dev.null.org/dadaengine/.

[4] S. F. Chen and J. T. Goodman, ‘An empirical study of smoothing techniques for language modeling’, in *34th ACL*, pp. 310–318, Santa Cruz, (1996).

[5] D. Fetterly, M. Manasse, and M. Najork, ‘Spam, damn spam, and statistics: using statistical analysis to locate spam web pages’, in *WebDB ’04*, pp. 1–6, New York, NY, USA, (2004).

[6] D. Fetterly, M. Manasse, and M. Najork, ‘Detecting phrase-level duplication on the world wide web’, in *ACM SIGIR*, Salvador, Brazil, (2005).

[7] Z. Gyöngyi and H. Garcia-Molina, ‘Web spam taxonomy’, in *AIRWeb Workshop*, (2005).

[8] Z. Gyöngyi., H. Garcia-Molina, and J. Pedersen, ‘Combating Web spam with TrustRank’, in *VLDB’04*, pp. 576–587, Toronto, Canada, (2004).

[9] P. Heymann, G. Koutrika, and H. Garcia-Molina, ‘Fighting spam on social web sites: A survey of approaches and future challenges’, *Internet Computing, IEEE*, **11**(6), 36–45, (2007).

[10] A. Kołcz and A. Chowdhury, ‘Hardening fingerprinting by context’, in *CEAS’07*, Mountain View, CA, USA, (2007).

[11] T. Lavergne, ‘Taxonomie de textes peu-naturels’, in *JADT’O8*, volume 2, pp. 679–689, (2008). in French.

[12] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, MA, 1999.

[13] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, ‘Detecting spam web pages through content analysis’, in *WWW Conference*, (2006).

[14] K. Seymore and R. Rosenfeld, ‘Scalable backoff language models’, in *ICSLP ’96*, volume 1, pp. 232–235, Philadelphia, PA, (1996).

[15] V. Siivola and B. Pellom, ‘Growing an n-gram model’, in *9th INTERSPEECH*, pp. 1309–1312, (2005).

[16] B. Stein, S. Meyer zu Eissen, and M. Potthast, ‘Strategies for retrieving plagiarized documents’, in *ACM SIGIR*, pp. 825–826, New York, NY, USA, (2007).

[17] A. Stolcke. Entropy-based pruning of backoff language models, 1998.

[18] A. Stolcke. Srilm – an extensible language modeling toolkit, 2002.

[19] A. Franz T. Brants. Web 1T 5-gram corpus version 1.1, 2006. LDC ref: LDC2006T13.

[20] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne, ‘Tracking web spam with HTML style similarities’, *ACM TWeb*, **2**(1), 1–28, (2008).