Workshop Proceedings

ECAI Patras 2008

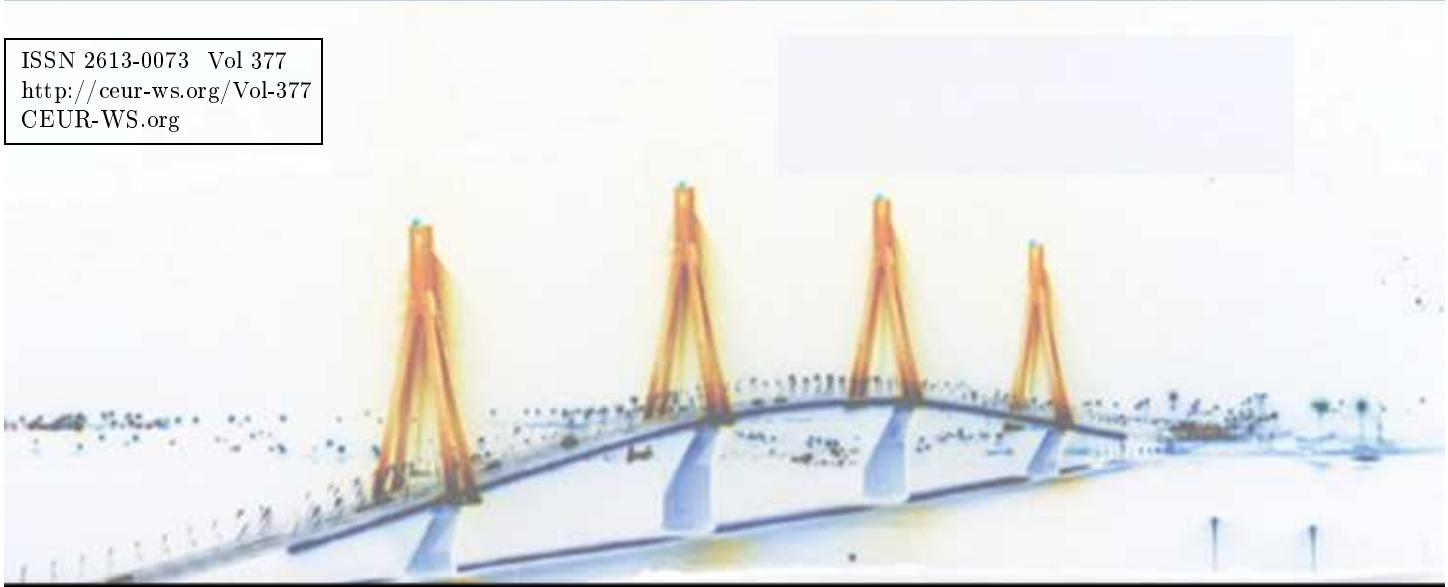
18th European Conference on Artificial Intelligence

22 July 2008

Uncovering Plagiarism, Authorship and Social Software Misuse - PAN'08

Benno Stein Efstathios Stamatatos Moshe Koppel

ISSN 2613-0073 Vol 377
http://ceur-ws.org/Vol-377
CEUR-WS.org



University of Patras

Preface

Plagiarism analysis is a collective term for computer-based methods to identify a plagiarism offense. In connection with text documents we distinguish between corpus-based and intrinsic analysis: the former compares suspicious documents against a set of potential original documents, the latter identifies potentially plagiarized passages by analyzing the suspicious document with respect to changes in writing style.

Authorship identification divides into so-called attribution and verification problems. In the authorship attribution problem, one is given examples of the writing of a number of authors and is asked to determine which of them authored given anonymous texts. In the authorship verification problem, one is given examples of the writing of a single author and is asked to determine if given texts were or were not written by this author. As a categorization problem, verification is significantly more difficult than attribution. Authorship verification and intrinsic plagiarism analysis represent two sides of the same coin.

Social software such as Web logs, sharing sites for photos and videos, wikis and on-line forums contribute up to one third of new Web content. "Social Software Misuse" is a collective term for anti-social behavior in online communities; an example is the distribution of spam via the e-mail infrastructure. Interestingly, spam is one of the few misuses for which detection technology is developed at all, though various forms of misuse exist that threaten the different online communities. Our workshop shall close this gap and invites contributions concerned with all kinds of social software misuse.

The workshop shall bring together experts and researchers around the exciting and future-oriented topics of plagiarism analysis, authorship identification, and the detection of social software misuse. The development of new solutions for these problems can benefit from the combination of existing technologies, and in this sense the workshop provides a platform that spans different views and approaches.

Benno Stein
Efstathios Stamatatos
Moshe Koppel

Program Committee

Shlomo Argamon, Illinois Institute of Technology, USA
Yaniv Bernstein, Google, Switzerland
Fazli Can, Bilkent University, Turkey
Carole Chaski, Institute for Linguistic Evidence, USA
Christian Gütl, University of Technology Graz, Austria
Graeme Hirst, University of Toronto, Canada
Heiko Holzheuer, Lycos Europe, Germany
Hans Kleine-Büning, University of Paderborn, Germany
Moshe Koppel, Bar-Ilan University, Israel
Hermann Maurer, University of Technology Graz, Austria
George Mikros, National and Capodestrian University of Athens, Greece
Sven Meyer zu Eissen, Bauhaus University Weimar, Germany
Martin Potthast, Bauhaus University Weimar, Germany
Paolo Rosso, Universidad Politecnica de Valencia, Spain
Efstathios Stamatatos, University of the Aegean, Greece
Benno Stein, Bauhaus University Weimar, Germany
Özlem Uzuner, State University of New York, USA

Content

On Cross-lingual Plagiarism Analysis using a Statistical Model

Alberto Barrón-Cedeño and **Paolo Rosso** and **David Pinto** and **Alfons Juan** [1]

Abstract. The automatic detection of plagiarism is a task that has acquired relevance in the Information Retrieval area and it becomes more complex when the plagiarism is made in a multilingual panorama, where the original and suspicious texts are written in different languages. From a cross-lingual perspective, a text fragment in one language is considered a plagiarism of a text in another language if their contents are considered semantically similar no matter they are written in different languages and the corresponding citation or credit is not included.

Our current experiments on cross-lingual plagiarism analysis are based on the exploitation of a statistical bilingual dictionary. This dictionary is created on the basis of a parallel corpus which contains original fragments written in one language and plagiarised versions of these fragments written in another language.

The process for the automatic plagiarism analysis based on the statistical bilingual dictionary has shown good results in the automatic cross-lingual plagiarism analysis and we consider that it could be useful for the cross-lingual near-duplicate detection task.

1 INTRODUCTION

Nowadays people enjoy an easy access to a wide range of information in multiple languages via the World Wide Web. Unfortunately, this "free access" to the information has caused a big temptation: the plagiarism, also from one language to another one. In some way, cross-lingual plagiarism analysis is related to cross-lingual information retrieval [6, 4]. In fact, the aim is to retrieve those fragments that have been plagiarised in a language with respect to the one originally employed.

In this paper we present an approach to the task of cross-lingual plagiarism analysis based on the exploitation of a statistical bilingual dictionary, commonly used in the automatic machine translation and cross-language infomation retrieval tasks [1, 4, 6].

The rest of the paper is organised as follows. Section 2 describes some of the current work in the task of cross-lingual plagiarism analysis. Section 3 introduces the definition and estimation process of the model used in order to create the statistical bilingual dictionary. Section 4 gives a description of the preliminary experiments carried out. Finally, Section 5 includes discussion and future work.

2 PRELIMINARY APPROACH(ES) IN CROSS-LINGUAL PLAGIARISM ANALYSIS

Some efforts have been made in other research directions that could be useful for this task. There have been developed, for example, some methods for the automatic acquisition of translated web pages [9], based on the search of hyperlinks containing strings of the kind "Spanish version" in order to download all the language versions of a given page. Although these cases cannot be considered plagiarism, the method could be useful in order to retrieve some instances for the training phase when dealing with cross language plagiarism analysis.

In [8] it has been proposed a method based on a thesaurus. In order to search document translations they have used the Eurovoc Thesaurus[2] to decide whether a document is near to another one in a different language or not. As the authors point out, this approach could be useful in the plagiarism analysis, of course if a good thesaurus is available.

The automatic plagiarism analysis may be classified into two main approaches: one with a reference corpus [10, 3] and one without it, which is also known as intrinsic plagiarism analysis [5, 11]. In the first case, the idea is to compare fragments (x_i) of a suspicious document (D_s) with fragments y_j of documents in a reference corpus (C) which is composed by original documents, in order to find those similar fragments that could be considered plagiarised. In the other case the objective is the same, but the idea is to look for variations through the text of the suspicious document (D_s) -like syntax, grammatical categories use and content complexity-, and do not exploit any reference corpus.

The state of the art in automatic plagiarism analysis allows to detect word by word plagiarism, even if fragments have been modified. However, to our knowledge, *Cross-Lingual Plagiarism Analysis* (CLiPA) nearly has been explored in the litherature.

The authors of [7] propose a method based on three main steps. Given a suspicious document d and a reference corpus C in a different language, the first step consists in retrieving a subset of candidate documents from C which could be sources of the plagiarised fragments of the document d. Then a semantic analysis is done between the sections of d and each $c_i \in C$. Finally, the similar sections are analysed in order to filter those cases where a proper citation has been made. Authors are currently working on the improvement of the analysis step.

3 THE STATISTICAL MODEL

In this section we describe the statistical model (Sub-section 3.1) and the Expectation Maximisation method for the estimation of the probabilities of the bilingual dictionary (Sub-section 3.2). This bilingual dictionary is the core of the CLiPA system used in this research work.

[1] Dpto. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, email: {lbarron, prosso, dpinto, ajuan}@dsic.upv.es

[2] http://europa.eu/eurovoc/

3.1 Model definition

Let $x_1, x_2, \cdots x_V$ be fragments conforming a suspicious text V in a certain language, and let $y_1, y_2, \cdots, y_W$ be a collection of W original fragments in a different language (the reference corpus). Let $\mathcal{X}$ and $\mathcal{Y}$ be their associated vocabularies, respectively.

Given the suspicious fragment $x_j \in V$, our objective is to find the most similar original fragment $y_k \in W$. The obtained relations, could be the original and plagiarised pairs. In order to do this, we have followed a probabilistic approach in which the most similar original fragment is computed as the most probable given x, i.e.,

$$y_i^*(x) = \underset{y=y_1,\cdots,y_W}{\operatorname{argmax}} p(y \mid x) \tag{1}$$

In this work, $p(y \mid x)$ is modelled by using the well-known IBM alignment model 1 (IBM-1) for statistical machine translation [1, 2]. This model assumes that each word in the reference segment y_k is *connected to exactly one word* in the suspicious fragment x_j. Also, it is assumed that y_k has an initial "null" word to link it to those words in x_j with no direct connexion.

Formally, a hidden variable $a = a_1 a_2 \cdots a_{|y|}$ is introduced in order to reveal, for each position i in y_j, the suspicious fragment word position $a_i \in \{0, 1, \ldots, |x|\}$ to which it is connected. Thus,

$$p(y \mid x) = \sum_{a \in \mathcal{A}(x,y)} p(y, a \mid x) \tag{2}$$

where $\mathcal{A}(x, y)$ denotes the set of all possible alignments between x and y. The *alignment-completed* probability $p(y, a \mid x)$ can be decomposed in terms of individual, y_k position-dependent probabilities as:

$$p(y, a \mid x) = \prod_{i=1}^{|y|} p(y_i, a_i \mid a_1^{i-1}, y_1^{i-1}, x) \tag{3}$$

$$= \prod_{i=1}^{|y|} p(a_i \mid a_1^{i-1}, y_1^{i-1}, x) p(y_i \mid a_1^{i}, y_1^{i-1}, x) \tag{4}$$

In the case of the IBM-1 model, it is assumed that a_i is uniformly distributed

$$p(a_i \mid a_1^{i-1}, y_1^{i-1}, x) = \frac{1}{|x| + 1} \tag{5}$$

and that y_i only depends on the query word to which it is connected

$$p(y_i \mid a_1^{i}, y_1^{i-1}, x) = p(y_i \mid x_{a_i}) \tag{6}$$

By substitution of (5) and (6) in (4); and thereafter (4) in (2), we may write the IBM-1 model as follows by some straightforward manipulations:

$$p(y \mid x) = \sum_{a \in \mathcal{A}(x,y)} \prod_{i=1}^{|y|} \frac{1}{(|x| + 1)} p(y_i \mid x_{a_i}) \tag{7}$$

$$= \frac{1}{(|x| + 1)^{|y|}} \prod_{i=1}^{|y|} \sum_{j=0}^{|x|} p(y_i \mid x_j) \tag{8}$$

Note that this model is governed only by a *statistical dictionary* $\mathbf{\Theta}$=$\{p(w|v)$, for all $v \in \mathcal{X}$ and $w \in \mathcal{Y}\}$. The model assumes that the order of the words in the suspicious fragment is not important. Therefore, each position in a original fragment is equally likely to be connected to each position in the suspicious one. Although this assumption is unrealistic in machine translation, we do *not* actually perform any translation and we consider that the IBM-1 model is well-suited for approaching cross-lingual plagiarism analysis.

3.2 Maximum Likelihood Estimation

It is not difficult to derive an Expectation-Maximisation (EM) algorithm to perform maximum likelihood estimation of the statistical dictionary with respect to a collection of training samples $(X, Y) = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. The *(incomplete)* log-likelihood function is:

$$L(\mathbf{\Theta}) = \sum_{n=1}^{N} \log \sum_{a_n} p(y_n, a_n | x_n) \tag{9}$$

with

$$p(y_n, a_n | x_n) = \frac{1}{(|x_n| + 1)^{|y_n|}} \prod_{i=1}^{|y_n|} \prod_{j=0}^{|x_n|} p(y_{ni} \mid x_{nj})^{a_{nij}} \tag{10}$$

where, for convenience, the alignment variable, $a_{ni} \in \{0, 1, \ldots, |x_n|\}$, has been rewritten as an indicator vector, $a_{ni} = (a_{ni0}, \ldots, a_{ni|x_n|})$, with 1 in the suspicious fragment position to which it is connected, and zeros elsewhere.

The so-called *complete* version of the log-likelihood function (9) assumes that the hidden (missing) alignments $a_1, \ldots, a_N$ are also known:

$$\mathcal{L}(\mathbf{\Theta}) = \sum_{n=1}^{N} \log p(y_n, a_n | x_n) \tag{11}$$

An initial estimate for $\mathbf{\Theta}$, $\mathbf{\Theta}^{(0)}$, is required for the EM algorithm to start. This can be done by assuming that the translation probabilities are uniformly distributed; i.e.,

$$p(w \mid v)^{(0)} = \frac{1}{|\mathcal{Y}|} \; \forall v \in \mathcal{X}, w \in \mathcal{Y} \tag{12}$$

After this initialisation, the EM algorithm maximises (9) iteratively, through the application of two basic steps in each iteration: the E(xpectation) step and the M(aximisation) step. At iteration k, the E step computes the expected value of (11) given the observed (incomplete) data, (X, Y), and a current estimation of the parameters, $\mathbf{\Theta}^{(k)}$. This reduces to the computation of the expected value of a_{nij}:

$$a_{nij}^{(k)} = \frac{p(y_{ni} \mid x_{nj})^{(k)}}{\sum_{j'} p(y_{ni} \mid x_{nj'})^{(k)}} \tag{13}$$

Then, the M step finds a new estimate of $\mathbf{\Theta}$, $\mathbf{\Theta}^{(k+1)}$, by maximising (11), using (13) instead of the missing a_{nji}. This results in:

$$P(w|v)^{(k+1)} = \frac{\sum_n \sum_{i=1}^{|y_n|} \sum_{j=0}^{|x_n|} a_{nij}^{(k)} \, \delta(y_{ni}, w) \, \delta(x_{nj}, v)}{\sum_{w'} \sum_n \sum_{i=1}^{|y_n|} \sum_{j=0}^{|x_n|} a_{nij}^{(k)} \, \delta(y_{ni}, w') \, \delta(x_{nj}, v)} \tag{14}$$

$$= \frac{\sum_n \frac{p(w \mid v)^{(k)}}{\sum_{j'} p(w \mid x_{nj'})^{(k)}} \sum_{i=1}^{|y_n|} \sum_{j=0}^{|x_n|} \delta(y_{ni}, w)\, \delta(x_{nj}, v)}{\sum_{w'} \left[\sum_n \frac{p(w' \mid v)^{(k)}}{\sum_{j'} p(w' \mid x_{nj'})^{(k)}} \sum_{i=1}^{|y_n|} \sum_{j=0}^{|x_n|} \delta(y_{ni}, w')\, \delta(x_{nj}, v) \right]} \quad (15)$$

for all $v \in \mathcal{X}$ and $w \in \mathcal{Y}$; where $\delta(a, b)$ is the Kronecker delta function; i.e., $\delta(a, b) = 1$ if $a = b$; 0 otherwise.

4 Preliminary experiments

We have carried out some preliminary experiments by selecting five document fragments from one author of the information retrieval area. The aim of this experiment was to obtain a personalised bilingual statistical dictionary which may be used to perform an author-focused CLiPA. The five original fragments $y_{\{1\cdots5\}}$ are the following:

y_1 *Plagiarism analysis is a collective term for computed-based methods to identify a plagiarism offence. In connection with text documents we distinguish between corpus-based and intrinsic analysis: the former compares suspicious documents against a set of potential original documents, the latter identifies potentially plagiarised passages by analysing the suspicious document with respect to changes in writing style.*

y_2 *Plagiarism is the practice of claiming, or implying, original authorship of someone elses written or creative work, in whole or in part, into ones own without adequate acknowledgement.*

y_3 *A cluster algorithm takes a set D of objects as input and operationalizes a strategy to generate a clustering C. Informally stated, the overall objective of a cluster algorithm is to maximise the innercluster similarity and to minimise the intra-cluster similarity.*

y_4 *Near-duplicate detection is mainly a problem of the World Wide Web: duplicate Web pages increase the index storage space of search engines, slow down result serving, and decrease the retrieval precision*

y_5 *Intrinsic plagiarism analysis deals with the detection of plagiarised sections within a document d, without comparing d to extraneous sources*

For each original text fragment, we have constructed plagiarised cases based on two approaches: Machine Translation (MT) and Human Simulated (HS). In the former approach, we have used five popular on-line translators[3]

, whereas for the latter nine different people have "plagiarised" each original fragment written in English to fragments in Spanish.

In order to show the similarity between the plagiarised fragments based on the human process and on automatic machine translation, we show the Jaccard distance of the MT and HS pairs of plagiarised fragments in Tables 1 and 2, corresponding to the original fragments y_1 and y_3, respectively.

In both tables we can see that there is an important difference between MT and HS plagiarisms. Additionally, considering only one row, t_1 from Table 1 for example, we can see that there are significant differences between the HS plagiarisms simply considering the Jaccard distance with respect to t_1 or any of the other MT plagiarised fragments. The same behaviour fact can be appreciated for the MT plagiarisms fixing one HS column.

[3] Freetranslation ($www.freetranslation.com$)
Google ($www.google.com/language\_tools$)
Worldlingo ($www.worldlingo.com$)
Systran ($www.systransoft.com$)
Reverso ($www.reverso.net$)

Table 1. Jaccard distance (J_δ) for human plagiarised (h_i) and machine translation (t_j) fragments for y_1

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9
t_1	0.50	0.58	0.58	0.50	0.52	0.74	0.58	0.44	0.37
t_2	0.54	0.59	0.48	0.45	0.52	0.73	0.52	0.41	0.41
t_3	0.51	0.60	0.60	0.54	0.51	0.75	0.61	0.45	0.43
t_4	0.56	0.64	0.60	0.56	0.54	0.76	0.63	0.48	0.43
t_5	0.67	0.74	0.73	0.70	0.66	0.78	0.67	0.65	0.63

Table 2. Jaccard distance (J_δ) for human plagiarised (h_i) and machine translation (t_j) fragments for y_3

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9
t_1	0.47	0.70	0.59	0.55	0.48	0.69	0.56	0.49	0.53
t_2	0.46	0.64	0.56	0.51	0.53	0.68	0.57	0.48	0.55
t_3	0.49	0.72	0.58	0.54	0.56	0.71	0.60	0.54	0.58
t_4	0.49	0.69	0.56	0.51	0.56	0.71	0.60	0.51	0.55
t_5	0.64	0.81	0.67	0.66	0.62	0.84	0.70	0.66	0.61

In general, the complete corpus is made up of the following text fragments:

i Five original fragments written in English by a unique author
ii Nine human simulated plagiarisms for each original fragment
iii Five automatic machine translations for each original fragment
iv Fourty six unplagiarised (independent) fragments about the plagiarism topic originally written in Spanish language

We have splitted the complete corpus into two datasets: training and test. The training dataset, which is used to construct the statistical bilingual dictionary, is made up of 50 pairs composed of original fragments and their corresponding plagiarised versions. The plagiarised versions were those obtained by 3 MT and 7 HS plagiarisms. In the test dataset, we employed 46 Unplagiarised Text Fragments (UTF) distributed as follows: 20 text fragments obtained by rewriting the same original concept, but mostly with other words (UTF_1), and 26 text fragments without any relation with the one of the original text fragments (UTF_2)[4].

In order to verify the similarity among the text fragments of the test dataset, we have represented each text by using the vector space model and, thereafter, we calculated the cosine of the similarity among them. We were particularly interested in observing the similarity arithmetic mean obtained only among the Plagiarised Text Fragments (PTF), of the same original fragment. This in order to confirm that those texts are similar enough and, at the same time, they are different enough to be considered as a challenge. Moreover, we also calculated the arithmetic mean of the similarity between the plagiarised vs. unplagiarised text fragments. The obtained results are shown in Table 3. We may observe that the plagiarised documents obtained an average of 0.44 which we consider to be good for the purposes of this preliminary investigation. The unplagiarised documents obtained instead, very low average of similarity.

We compute the probability $p(y|x)$ of each original text fragment given a suspicious one of the test subcorpus, on the basis of the statistical bilingual dictionary that we have obtained during the training phase (Section 3). The entire process of the experiment is illustrated in Figure 1.

We have conducted experiments in order to define the number of necessary iterations of the EM algorithm. We have calculated bilingual dictionaries with $k = \{10, 20, \cdots, 100\}$ where k is the number

[4] The CLiPA corpus is freely available at www.dsic.upv.es/grupos/nle/downloads.html

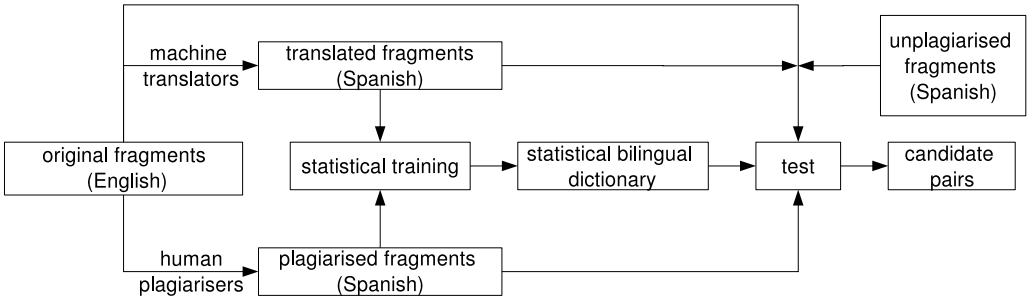


Figure 1. Experiment description (including training and test)

Table 3. Analysis of similarities for the test dataset

Fragment-Fragment	Similarity average	Minimum similarity	Maximum similarity
PTF-PTF	0.447	0.153	0.929
PTF-UTF_1	0.089	0.002	0.344
PTF-UTF_2	0.028	0.002	0.133

of iterations in the EM algorithm. We are only interested in defining a good number of iterations for the training phase. Due to this fact, we only consider that a fragment x_i has been recognised if it fulfills the following condition:

$$p(y_i|x_i) > p(y_j|x_i) \ \forall j \neq i \tag{16}$$

We have used a variation of the Precision measure: *Precision at 1* (P@1)[5]. Figure 2 shows the behaviour of P@1 for different EM iterations. In agreement with [6], we have considered a maximum number of 100 iterations in order to avoid over-training.

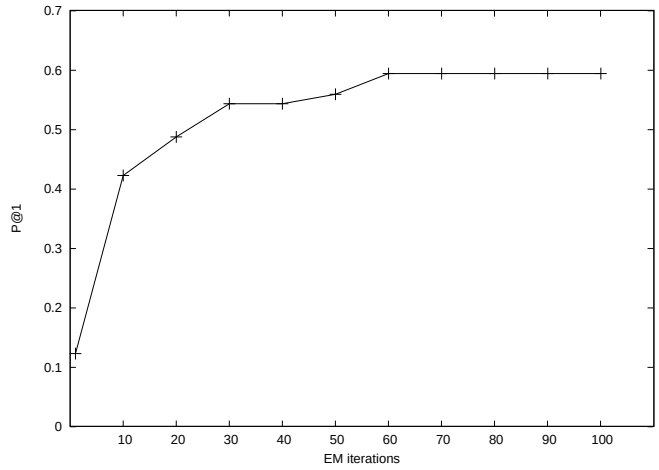


Figure 2. P@1 with different EM iterations

[5] In P@1 only the best ranked item in the output is considered for the precision calculation.

In Figure 2 we can observe that the P@1 value reaches a certain stability after 60 iterations. The preliminary results are interesting and they encourage to further continue in this research direction. However, the results need to be validated in the future on a bigger corpus.

In order to obtain a discriminate the good candidates, we have tested different threshold values. Figure 3 shows the behaviour of the F-measure based on different thresholds. The curve in this figure must be analysed from right to left. In the highest values of the threshold, the F-measure is low due to the fact that Recall is near to zero. Meanwhile the threshold descends, more actual plagiarised fragments are considered and the F-measure is incremented. The best value is obtained when $Threshold = 4.31 * 10^{-9}$ where a good part of the real plagiarised fragments have a probability of being detected as plagiarised that is little higher than the threshold value. After this peak, both precision and F-measure decrease. This is the reason why we opted for using this value as threshold for which $P = 1$ and $R = 0.8$ (F-measure=0.88).

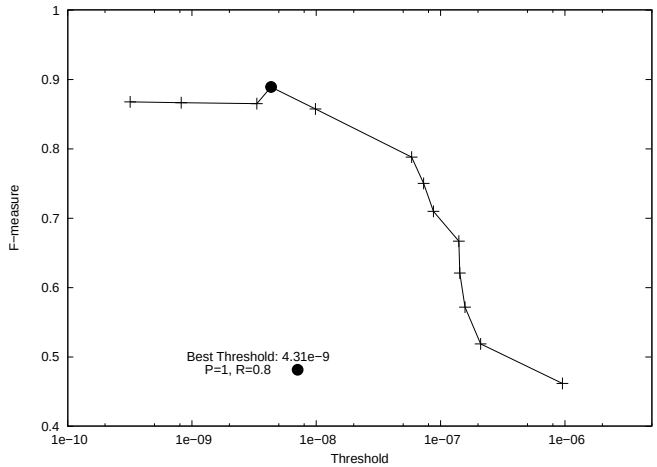


Figure 3. F-measure for different threshold values

In order to clarify the obtained results, we consider the following three Spanish text fragments. The first two are examples of plagiarised fragments in Spanish of y_5 and y_1, respectively. The third one

is an example of unrelated text.

x_1 *El análisis del plagio intrinseco tiene que ver con la detección de secciones plagiadas de un documento d, sin comparar d con fuentes externas*

x_2 *El analisis del plagio es un término colectivo para que los metodos computar-basados identifiquen una ofensa del plagio. Con respecto a documentos del texto distinguimos entre el análisis recopilación-basado e intrínseco: el anterior compara documentos sospechosos contra un sistema de documentos originales potenciales, el último identifica pasos potencialmente plagiados analizando el documento sospechoso con respecto a cambios en estilo de escritura.*

z_1 *Hipótesis La perplejidad de un fragmento perteneciente a un escritor con respecto a otro, será mayor que la de dos documentos escritos por el mismo autor. Aquellos párrafos que tengan mayor perplejidad sera los mejores candidatos a ser fragmentos plagiados.*

x_1 is one case of a HS plagiarism. One translation of this fragment could be "*Intrinsic plagiarism analysis has to do with the detection of plagiarised sections from a document d without comparing d to external sources*" and, obviously, is a plagiarism of y_5. In this case $p(y_5|x_1) = 33.1 \cdot 10^{-5}$ which exceeds the previously defined threshold and, therefore, x_1 is considered a plagiarism of y_5. x_2 has been generated from y_1 by using an on-line machine translator. In this case $p(y_1|x_2) = 10.28 \cdot 10^{-9}$ and, therefore, x_2 is considered a plagiarism of y_1.

With respect to z_1, $p(y_i|z_1) \approx 0$ and, therefore, z_1 is not considered to be a plagiarism of any original text fragment of the reference corpus. For instance, the following words *hipótesis*, *párrafos*, *perplejidad* and *mejores* (hypothesis, paragraphs, perplexity and best) do not have any relation with the English words in the reference corpus and, therefore, the association probability between them is close to zero.

5 CONCLUSIONS

In this paper we have approached the cross-lingual plagiarism analysis with a probabilistic method which calculates a bilingual statistical dictionary on the basis of the IBM-1 model. In order to generate the bilingual dictionary, we have used a set of original documents written in English and Spanish plagiarised examples. Our proposal calculates the probabilistic association between two terms in two different languages. The main contribution of this paper is that the probabilistic model is trained with a data set made of pairs of fragments of text from a particular author. The aim of our approach is to investigate the cross-lingual plagiarism with respect to a specific author.

The application of a statistical machine translation technique (without any translation), has demonstrated to be a valuable resource for the CLiPA task. Due to the fact that we determine the similarity between suspicious and original text fragments based on a dictionary, the order of the words in the fragment is not relevant and we are able to find good candidates even when the plagiarised text has been modified.

We believe that this approach is not only useful for the cross-lingual plagiarism analysis, but for the near-duplicate analysis too. As further work, we would like to validate the results we obtained in this preliminary experiment on a bigger corpus. Unfortunately, the construction of a cross-lingual corpus with the required characteristics, in size and quality, seems to be by itself a sufficiently difficult task which makes cross-language plagiarism analysis even more challenging.

References of the Original Text Fragments

y_1 Preface of the Proc. of the International Workshop on Plagiarism Analysis, Authorship Identification , and Near-Duplicate Detection (PAN 2007).

y_2 Introduction of the lecture on "Technology for Plagiarism Analysis" given by Benno Stein at the UPV in March of 2008.

y_3 B. Stein and M. Busch. 'Density-based Cluster Algorithms in Low-dimensional and High-dimensional Applications'. Stein and Meyer zu Eissen (eds.), 2nd Int. Workshop on Text-Based Information Retrieval (TIR 05), *Germany*, 45-56, (2005).

y_4 See reference of y_1.

y_5 B. Stein and S. Meyer zu Eissen. 'Intrinsic Plagiarism Analysis with Meta Learning'. SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN 07), *Netherlands*, 45-50, (2007).

ACKNOWLEDGEMENTS

We would like to thank the MCyT TIN2006-15265-C06-04 research project for partially funding this work as well as to the CONACyT-MEXICO 192021 grant and the BUAP-701 PROMEP/103.5/05/1536 grant.

REFERENCES

[1] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, 'A statistical approach to machine translation', *Computational Linguistics*, **16**(2), 79–85, (1990).

[2] J. Civera and A. Juan, 'Mixtures of ibm model 2', *Proc. of the EAMT Conference*, 159–167, (2006).

[3] Parvati Iyer and Abhipsita Singh, 'Document similarity analysis for a plagiarism detection system', *2nd Indian Int. Conf. on Artificial Intelligence (IICAI-2005)*, 2534–2544, (2005).

[4] W. Kraaij, J. Y. Nie, and M. Simard, 'Embedding web-based statistical translation models in cross-language information retrieval', *Computational Linguistics*, **29**(3), 381–419, (2003).

[5] Sven Meyer zu Eissen and Benno Stein, 'Intrinsic plagiarism detection', *Lalmas et. al. (Eds.): Advances in Information Retrieval Proc. of the 28th European Conf. on IR research, ECIR 2006, London*, 565–569, (2006).

[6] David Pinto, Alfons Juan, and Paolo Rosso, 'Using query-relevant documents pairs for cross-lingual information retrieval', *TSD 2007. Springer-Verlag*, **LNAI (4629)**, 630–637, (2007).

[7] Martin Potthast, Benno Stein, and Maik Anderka, 'A wikipedia-based multilingual retrieval model', *Macdonald, Ounis, Plachouras, Ruthven and White (eds.). 30th European Conf. on IR Research (ECIR 2008), Glasgow, Springer-Verlag*, **LNCS (4956)**, 522–530, (2008).

[8] Bruno Pouliquen, Ralf Steinberger, and Camelia Ignat, 'Automatic identification of document translations in large multilingual document collections', *Proc of the Int. Conf. Recent Advances in Natural Language Processing (RANLP-2003), Borovets, Bulgaria*, 401–408, (2003).

[9] Philip Resnik, 'Mining the web for bilingual text', *37th Annual Meeting of the Association for Computational Linguistics (ACL 99), Maryland*, (1999).

[10] Antonio Si, Hong Va Leong, and Rynson W. H. Lau, 'Check: a document plagiarism detection system', *Proc. of the 1997 ACM Symposium on Applied Computing, San Jose, California, United States*, 70–77, (1997).

[11] Benno Stein and Sven Meyer zu Eissen, 'Intrinsic plagiarism analysis with meta learning', *B. Stein, M. Koppel, and E. Stamatatos, Eds., SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN 07), Amsterdam, Netherlands*, 45–50, (2007).

Towards the Exploitation of Statistical Language Models for Plagiarism Detection with Reference

Alberto Barrón-Cedeño and **Paolo Rosso**[1]

Abstract. To plagiarise is to robe credit of another person's work. Particularly, plagiarism in text means including text fragments (and even an entire document) from an author without giving him the correspondent credit. In this work we describe our first attempt to detect plagiarised segments in a text employing statistical Language Models (LMs) and perplexity.

The preliminary experiments, carried out on two specialised and literary corpora (including original, part-of-speech and stemmed versions), show that perplexity of a text segment, given a Language Model calculated over an author text, is a relevant feature in plagiarism detection.

1 INTRODUCTION

The Automatic Plagiarism Detection, a close related problem to the Automatic Authorship Attribution, has became a relevant task in Information Retrieval, scholar environments and even scientific circles.

There are some applications which try, for example, to detect whether a student report is plagiarised or not [2]. However, inside of specialised circles, there are cases when a person takes text fragments from other authors without making the corresponding citation and, in extreme cases, different authors claim for the authorship of a text and even an idea.

Language Models, commonly used in Speech Recognition [7] and Information Retrieval [11, 5], have been exploited in Automatic Authorship Attribution of text [10, 2] and even of source code [4]. In the first case, character level n-grams and perplexity are considered to determine the authorship of the analysed document. In the second case, frequencies of byte level n-grams are used to decide.

State of the art Automatic Plagiarism Detection allows to detect word by word plagiarism, even when fragments have been modified [14, 6]. In this work we are trying to exploit lexical and grammatical level Language Models (n-grams and perplexity) to detect plagiarised fragments in a text.

The paper is organised as follows. Section 2 describes some of the current advances in the task of plagiarism detection with a reference corpus. Section 3 gives an overview of statistical Language Models and perplexity, in order to determine how well a Language Model could represent a language. Section 4 gives a description of the preliminary experiments we carried out over specialised and literary texts (Sections 4.1 and 4.2) and discusses the obtained results (Section 4.3). Finally, in Section 5 we draw some conclusions.

[1] Dpto. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, email: {lbarron, prosso}@dsic.upv.es

[2] See for instance http://www.turnitin.com/static/plagiarism.html

2 CURRENT APPROACHES IN AUTOMATIC PLAGIARISM DETECTION WITH REFERENCE

The automatic plagiarism detection can be mainly classified in two approaches based on the exploitation (or not) of a reference corpus.

In the case when no reference corpus is exploited [9, 16], the idea is to find variations through the text of the suspicious document (D_s), like syntax, grammatical categories, text complexity or the verbal form (*I play*, *she plays*, *we played*) used in the text. On the other hand, when a reference corpus is used [14, 6], the basic idea is to compare fragments (f) of the suspicious document (D_s) with the documents in a reference corpus (C). Of course, the reference corpus contains only non-plagiarised documents.

The reason for using a reference corpus in order to detect plagiarism in a given text is obvious. In order to decide if a text is plagiarised, we should compare it with other texts looking for common fragments.

In this way, the task could be reduced to make an exhaustive comparison to answer the question: *is there a fragment $f \in D_s$ included in a document of C?*

If this problem is approached directly, two difficulties appear immediately; the first one is the need of a huge big reference corpus in order to make a serious search of fragments $f \in D_s$ in C, and second, the processing cost of making all the necessary comparisons is, in a high level, $O(n \cdot m)$ being n and m the length of D_s and C in fragments respectively (the real cost of this kind of comparisons decreases dramatically using hash-based techniques [15]).

Trying to avoid these difficulties the CHECK system [14] preprocesses the documents to determine their "semantic meanings", considering factors like document structure or keywords. This system detects the subject of D_s in order to only compare it with the related documents in C, the original documents corpus. In those cases where paragraphs in D_s and C are semantically related, a per-sentence comparison is made.

The same CHECK architecture is used in [6], but the per-sentence comparison is made using the *dot plot* technique. The advantage is that each word in the analysed sentence is compared with all the words of the sentences in the reference corpus. Two sentences are considered similar if they pass a given threshold (based on the common occurrence of words), a reason to consider a sentence suspicious.

As we have said, the dimension of a corpus must be really big. For example, the plagiarism detection tool offered by *Turn it in* (see footnote in Section 1) not only searches fragments in a reference corpus, but also in the Web.

3 On statistical Language Models

A statistical Language Model (*LM*) "tries to predict a word given the previous words" [8]. LMs have been mainly used in speech and optical character recognition [1, 12], and statistical machine translation [3, 17] between other Natural Language Process applications, but are not limited to these tasks.

To predict which word is the next given its history, the best option should be to consider all the words before it in the text. The probability of a given sentence $w_1 w_2 \dots w_n$, if we know $w_{\{1,2,\dots,n-1\}}$ but not w_n, would be given by the Bayes conditional probability, based on the chain rule, $P(W) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1 w_2) \cdots P(w_n|w_1 \cdots w_{n-1})$. Unfortunately, the training set to correctly define these probabilities must be extremely big and, no matter the extension, we will never have a representation for all the possible sentences in a text.

The option is to consider LMs only of n-grams. Over this framework, the model is based on strings conformed by n words, including the analysed one (common values are $n = \{2, 3\}$). The n-gram probability definition considering, for example, $n = 3$ is $P_3(W) = P(w_{n-2}) \cdot P(w_{n-1}|w_{n-2}) \cdot P(w_n|w_{n-2}w_{n-1})$.

Our main idea is that if we compute the probability of n-grams in a corpus of texts from one author, we will have a representation of her vocabulary, grammatical frequency and even writing style. These representations can be compared to other texts in order to look for candidates for plagiarised segments.

The question now is how to determine if a text is similar to another one. Alike [10], we have opted for perplexity, one way to express language theory's entropy, that is frequently used in order to evaluate how good a LM describes a language: "our author language".

Formula 1 includes the equation of perplexity (PP), where N is the number of tokens in the analysed text and $P(w_i|w_i - 1)$ is the probability of word w_i given w_{i-1}. This is the case for the perplexity calculation for bigrams.

$$PP = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_i - 1)}} \quad (1)$$

The lower a text perplexity is, the more predictable its words are. In other words, the higher a perplexity is, the bigger the uncertainty about the following word in a sentence ([8, pp. 60-78] to further investigate this issue).

Two tools for LMs and perplexity calculation freely availabe are SRILM and Cambridge-CMU [3]. In the preliminary experiments we carried out the first one.

4 THE LANGUAGE MODEL APPROACH

No matter there exist works where multiple features are considered for the task of automatic plagiarism detection (such as [9]), we have opted for starting our explorations in this area working only with one feature: perplexity. It is right for this reason that our results cannot be directly compared with others obtained from more robust techniques.

At the moment our aim is not to improve current results in this field, but to determine whether this kind of characterisation of an author style could be useful for this task or not in order to possibly combine it with other features in the future.

As we have seen in Section 3, a low perplexity means that, given a sequence of words, we are prepared enough to predict, with a low error rate, which will be the next one. Considering this, we define our main hypothesis:

Hypothesis Let k be the LM of a corpus composed by texts T written by an author A. The perplexity of fragments $g, h \in T'$, given that the fragment g has been written by A and the fragment h has been "plagiarised" from another author will be clearly different. Specifically, $PP_k(g) \ll PP_k(h)$.

Trying to prove (or reject) our hypothesis, we have carried out two main experiments: one over "specialised texts" (scientific papers) and another over "general literature texts" (novels, child literature), which we describe in Sections 4.1 and 4.2, respectively.

For these experiments, we have not only used the original documents. We have pre-processed all the texts in order to consider:

i the original text
ii the part-of-speech of the text
iii the stemmed text

We consider these three versions of the text in order to be able to represent the writer style. Specifically, we tried to recognise author's vocabulary and syntactic richness, (*i*) and (*iii*), and morphosyntactic style (*ii*). Part-of-speech and stems have been obtained with Treetagger [13].

Independent LMs have been calculated over the three versions of the training corpus considering $\{2 - 4\} - grams$.

With respect to the testing, we split the test corpus in sentences including those that were "artificially plagiarised" before applying to them the same pre-processing that to the training set (we have considered the dot as the only delimiter between them).

4.1 Experiments over specialised texts

For this case, we have used a corpus about Lexicography topics written by only one author. One section of the corpus (composed of around 11,628 words), was used for LM calculation and the other one for test. In the test partition, we randomly inserted fragments about related topics, but written by other authors.

In order to identify the "plagiarised" fragments (in this case paragraphs), we calculated the perplexities of each sentence with respect to the LM of the author. Figure 1 shows the perplexity of each sentence in the test corpus based on trigrams[4].

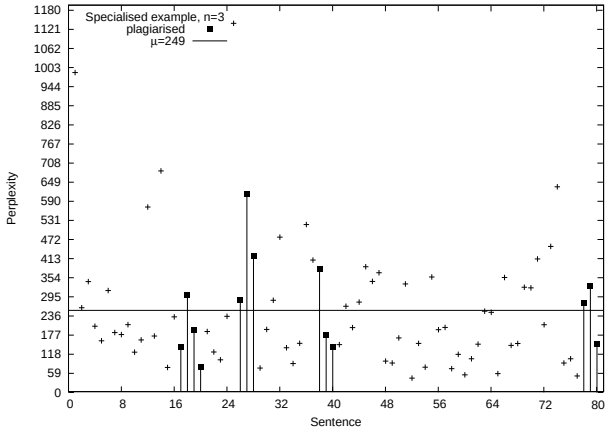
Due to the fact that it considers aspects such as singular/plural and verbal time, the perplexity values of the original text (a) are the highest of the three. The highest perplexities are $PP_{25} = 1132.15$ and $PP_1 = 980$, where 25 and 1 are the number of the sentence in the entire text. Sentence S_{25} has only seven words and contains a cite of the kind "$author_a$ *(2001)*" and $author_a$ did not appear in the training corpus, therefore, probability $P(author_a \in n - gram) \rightarrow 0$. In the case of sentence S_1, it contains the title of the paper, author and author's organisation, that is not English, so it contains words in another language.

The first plagiarised segment appears in the sixth place of the list of sentences sorted by perplexity. It is S_{27}: "*Such plain text representation is usually processed to add structure explicitly in a machine readable form.*" with $PP_{27} = 608.21$. This sentence contains six words that never appeared in the training corpus.

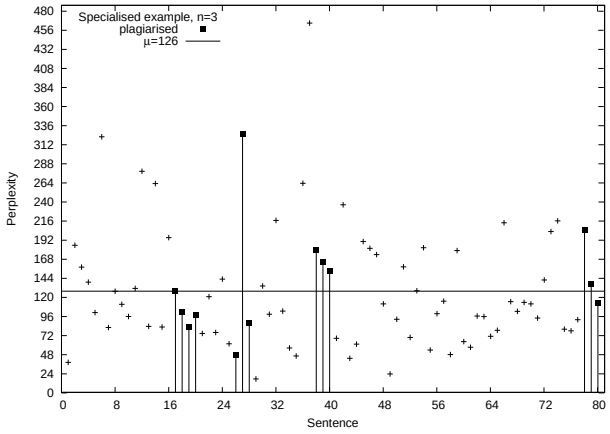
Working on the stemmed text (b) we consider only the richness of author's vocabulary without caring about the additional features

[3] See $http : //www.speech.sri.com/projects/srilm/$ and $http : //www.speech.cs.cmu.edu/SLM_info.html$ respectively

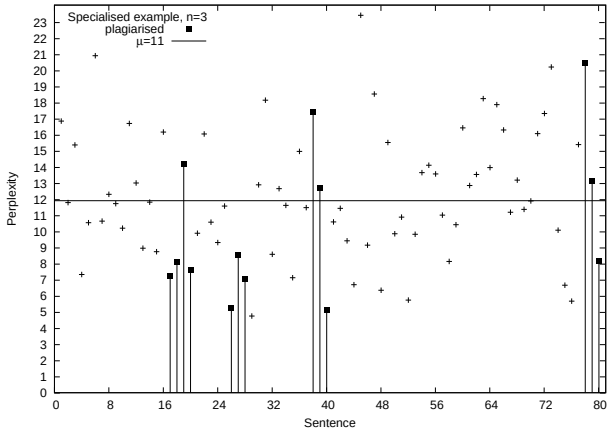
[4] In Figures 1 and 2 symbol "+" represents non-plagiarised sentences and a black square with a vertical bar plagiarised ones.



(a) original



(b) stemmed



(c) POS

Figure 1. Perplexity on the specialised corpus (one point per sentence)

considered in the original text. The highest perplexities in this case are $PP_{37} = 462.78$ and $PP_{27} = 323.46$. Sentence S_{37} is a fragment copied by the author from another text in order to analyse it and, therefore, the result can be considered correct. Sentence S_{27}, as we have already said is plagiarised.

Finally, in the part-of-speech version of the text (c) the vocabulary is clearly smaller than in the other two cases (around 40 words given by the grammar categories[5]), resulting in a perplexity range much lower. In this case the three highest perplexities are $PP_{45} = 23.36$, $PP_6 = 20.87$ and $PP_{78} = 20.40$. Between the twenty tokens in S_{45}, three are non-frequent strings conformed by parenthesis and cardinal numbers, for example *(2)* which is tagged like *(LS)* (list item). S_{78} is plagiarised and contains the 3-gram *DT NN IN*, which is the third trigram with smaller probability and others that have not appeared in the training corpus, which is the case of 3-grams *RB VVZ DT* and *DT RBR JJ* [6] and, therefore, their probability tends to 0.

These experiments have been made considering a little corpus. In Section 4.2 we describe the experiments we have carried out over a bigger corpus and, for this reason, a richer LM.

4.2 Experiments over general literature texts

In order to have a reference for our results, we have made the same experiments over a literary corpus. For these experiments we have taken a set of books written by the author Lewis Carroll and some passages from William Shakespeare texts to "plagiarise" the test section of Carroll's corpus[7]. The distribution of the training and test subcorpora is described in Table 1.

Table 1. Literary corpus

Author	Subcorpus	$\|w\|$
Carroll	training	116,202
Carroll	test	26,626
Shakespeare	plagiarised	103

We have done the same pre-process, described at the end of Section 4, to the training and test corpora in order to obtain original, part-of-speech and stem versions of the texts. Figure 2 shows the results over the three versions of the test corpus. In this case we can see that the plagiarised sections, in general, obtain high values of perplexity with respect to non-plagiarised segments in POS and stem versions.

However, it is clear in the three subfigures that non-plagiarised fragments have the highest perplexities. For example, in the case of the original text, the sentence with the highest perplexity, as it appears in the text, is "ALL **PERSONS** MORE **THAN** A **MILE** HIGH TO **LEAVE** THE **COURT**.". The words in bold have not appeared in the training corpus (at least with all the letters capitalised)[8].

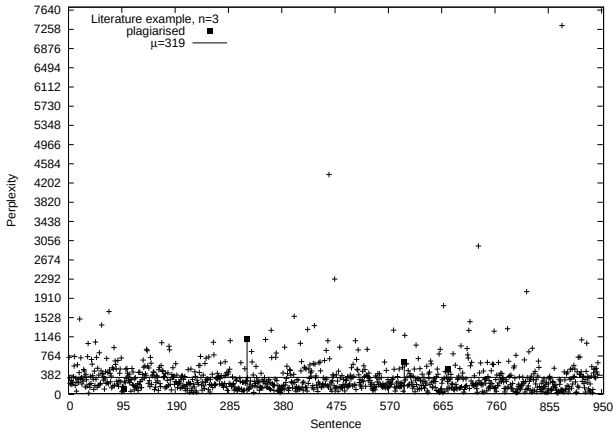
In the other two cases, the POS and stem versions, the reason for most of the cases is simple: there are errors in the part-of-speech and stems generated by the tagger (in some cases it is due to errors in the text). Let us consider the stemmed version of the test corpus to

[5] See http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/PennTreebank-Tagset.ps

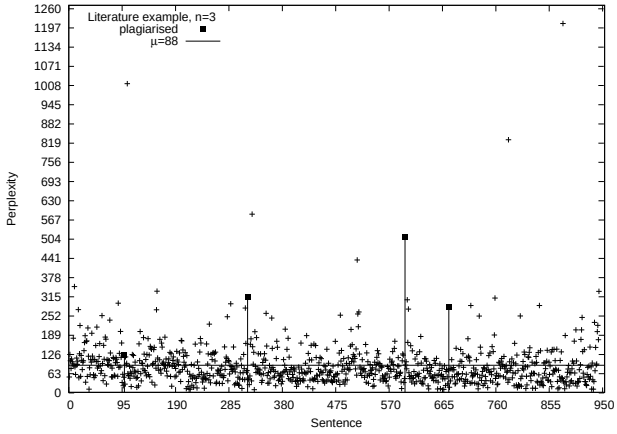
[6] DT=determiner; NN=noun; IN=preposition; RB=adverb; VVZ=verb; RBR=comparative adverb; JJ=adjective.

[7] Texts have been downloaded from Project Gutenberg, http://www.gutenberg.org

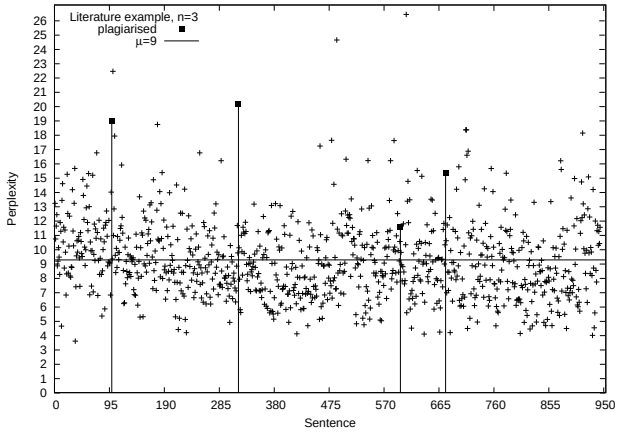
[8] These kind of "errors" could be solved converting all the characters to lower case during the pre-processing of the corpus.



(a) original



(b) stemmed



(c) POS

Figure 2. Perplexity on the literature corpus (one point per sentence)

show some examples. Table 2 includes the sentences with the highest perplexities in the Caroll's plagiarised document.

Table 2. Sentences with highest perplexities

Perplexity	Sentence
1205.6	all Persons more Than A Mile High TO leave the court.
1009.1	William 's conduct at first be moderate.
825.6	the twelve juror be all write very busily on slate.
582.5	' oh , there go his precious nose ' ; as an unusually large saucepan fly close by it , and very nearly carry it off.
508.1	the hearing of my wife , with your approach : so humbly take my

The first sentence contains proper nouns (capitalised words *Persons*, *Than* and *Mile* were all considered proper nouns by the tagger), which are hard to occur in different texts and, in this case, have $P(w) \rightarrow 0$ because they do not appear in the LM. This is one of the weaknesses of the original and stemmed versions of texts: due to the fact that they have an open language, it is difficult that a LM contains all the "strange words" as it is the case of proper names and other "special" words that are commonly included in texts.

All the words in the second sentence were included in the training corpus and, therefore, the vocabulary in the two cases is not different. However, *William* never appeared at the beginning of a sentence and the trigram *William 's conduct* neither, just to give a couple examples. In the case of the third sentence, it contains the word *juror*, that the LM ignores, and *busily*, that has a low probability: $P('busily') = 0.0000191$ (in order to compare, $P('the') = 0.03869$).

The first plagiarised line is the fifth one. The interesting fact here is that the LM knows all the words in this phrase, but the words and 3-grams in it have a low probability.

In the case of the part-of-speech version, the sentence with the highest perplexity has $PP_{608} = 26.34$, and it contains, for example, the substring *DT NN RBR* (determiner, noun, comparative adverb). This POS trigram corresponds to the segment of three tokens $(that)_1\,(is - -''The)_2\,(more)_3$, which, due to an error in words split was not correctly tagged and the resulting POS trigram has a really low probability.

In this case, the first plagiarised sentence in the sorted list is in the fourth place with $PP_{318} = 20.132$. This sentence has style and vocabulary completely different from the others in the text and corresponds to the sentence "*Mac. We will proceed no further in this Businesse: He hath Honour'd me of late, and I haue bought Golden Opinions from all sorts of people, Which would be worne now in their newest glosse, Not cast aside so*", written by W. Shakespeare.

4.3 Discussion

From the five categories of stylometric features useful for the plagiarism detection task [9], our LM approach considers just three of them. *Syntactic features* and *special words counting*, "which measure writing style at the sentence-level" [9] and vocabulary richness respectively, are considered with the perplexity calculation of the sentences over the original and stemmed test corpora. *Part-of-speech classes quantifycation* is implicitly considered with the POS version. The only two features that our approach does not consider are *text statistics* (at character level) and *structural features*, that deal with the organisation of the text.

It can be seen in the results of the experiments in Sections 4.1 and 4.2 that considering only the perplexity of a sentence is not good

enough to discriminate it from a plagiarised or ”legal” text fragment.

Considering the perplexity calculations over the three versions of the text (original, POS and stem) have conducted to the detection of “non-expected“ sentences that, in the most of the cases, include those that have been plagiarised. However, these three experiments do not detect the same phrases, but different ones, so we believe that we need to consider the three versions together in order to detect plagiarised sentences.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have explored the utility of Language Models and perplexity, a measure to determine the coverage of a Language Model given a text, for the Automatic Detection of Plagiarism with a reference corpus. We have considered perplexity on three different levels: word, part-of-speech and stem.

In order to do that, we have calculated a Language Model over a reference corpus, written by one only author, and calculated perplexity of sentences on a test corpus (which could be plagiarised or not) based on this model.

Our main hypothesis was that those segments with the highest perplexities with respect to the model, should be the plagiarised ones. Unfortunately, our hypothesis is not completely true because there are non-plagiarised fragments (in particular those with ”strange segments” such as titles and bibliographic cites) that present high perplexity. However, plagiarised segments seem to stand out in the highest positions when we consider these features.

In the results that we have obtained, we have noted that in order to identify good candidates for plagiarised segments we should consider the three versions of the analysed text together (original, POS and stem).

We know that the perplexity feature space of plagiarised and non-plagiarised segments is not completely separable, but we believe that including perplexity between other features may improve the results.

ACKNOWLEDGEMENTS

We would like to thank the MCyT TIN2006-15265-C06-04 research project for partially funding this work and the National Council for Science and Technology (CONACYT-MEXICO) for funding the research work of the first author (Id. 192021).

REFERENCES

[1] Thomas M. Breuel, ‘The ocropus open source ocr system’, *Proc. IS&T/SPIE 20th Annual Symposium 2008*, (2008).

[2] Rosa Maria Coyotl-Morales, Luis Villaseñor Pineda, Manuel Montes-y Gómez, and Paolo Rosso, ‘Authorship attribution using word sequences’, *Proc. of the 11th Iberoamerican Congress on Pattern Recognition, (CIARP 2006)*, **LNCS (4225)**, 844–853, (2006).

[3] Josep M. Crego, Jos B. Mariño, and Adriá de Gispert, ‘An n-gram-based statistical machine translation decoder’, *Interspeech’2005 - Eurospeech, Lisbon, Portugal*, (2005).

[4] Georgia Frantzeskou, Efstathios Stamatatos, and Stefanos Gritzalis, ‘Identifying authorship by byte-level n-grams: The source code author profile (scap) method’, *Int. Journal of Digital Evidence*, **6**(1), (2007).

[5] Djoerd Hiemstra, ‘A linguistically motivated probabilistic model of information retrieval’, *Proc. of the 2nd European Conference on Research and Advanced Technology for Digital Libraries, (ECDL 1998)*, **LNCS (1513)**, 569–584, (1998).

[6] Parvati Iyer and Abhipsita Singh, ‘Document similarity analysis for a plagiarism detection system’, *2nd Indian Int. Conf. on Artificial Intelligence (IICAI-2005)*, 2534–2544, (2005).

[7] Frederick Jelinek, *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, Massachusetts, 1997.

[8] Christopher D. Manning and Hinrich Schutze, *Foundations of Statistical Natural Language Processing*, The MIT Press Publisher, Cambridge Massachusetts and London, England, 2000.

[9] Sven Meyer zu Eissen and Benno Stein, ‘Intrinsic plagiarism detection’, *Lalmas et. al. (Eds.): Advances in Information Retrieval Proc. of the 28th European Conf. on IR research, ECIR 2006, London*, 565–569, (2006).

[10] Fuchim Peng, Dale Schuurmans, Vlado Keselj, and Shaojun Wang, ‘Automated authorship attribution with character level language models’, *Proc. of the 10th Conf. of the European Chapter of the Association for Computational Linguistics (EACL 2003), Budapest, Hungary*, (2003).

[11] Jay M. Ponte and W. Bruce Croft, ‘A language modeling approach to information retrieval’, *Croft, Moffat, van Rijsbergen, Wilkinson, and Zobel, Eds., 21st Annual Int. ACM SIGIR Conf.*, 275–281, (1998).

[12] Verónica Romero, Vicente Alabau, and José-Miguel Benedí, ‘Combination of n-grams and stochastic context-free grammars in an offline handwritten recognition system’, *3rd Iberian Conf. on Pattern Recognition and Image Analysis, Springer-Verlag, Girona, Spain*, **LNCS (4477)**, 467–474, (2007).

[13] Helmut Schmid, ‘Probabilistic part-of-speech tagging using decision trees’, *Proc. of the Int. Conf. on New Methods in Language Processing*, (1994).

[14] Antonio Si, Hong Va Leong, and Rynson W. H. Lau, ‘Check: a document plagiarism detection system’, *Proc. of the 1997 ACM Symposium on Applied Computing, San Jose, California, United States*, 70–77, (1997).

[15] Benno Stein, ‘Principles of hash-based text retrieval’, *Clarke, Fuhr, Kando, Kraaij, and de Vries, Eds., 30th Annual Int. ACM SIGIR Conf.*, 527–534, (2007).

[16] Benno Stein and Sven Meyer zu Eissen, ‘Intrinsic plagiarism analysis with meta learning’, *B. Stein, M. Koppel, and E. Stamatatos, Eds., SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN 07), Amsterdam, Netherlands*, 45–50, (2007).

[17] Richard Zens and Hermann Ney, ‘N-gram posterior probabilities for statistical machine translation’, *Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL): Proc. of the Workshop on Statistical Machine Translation, New York City, NY, United States*, (2006).

Pedigree Tracking in the Face of Ancillary Content

Eugene R. Creswick and **Emi Fujioka** and **Terrance Goan**[1]

Abstract. The accurate tracking and retrieval of content pedigree is a quickly growing requirement as our abilities to create information assets increases exponentially. Plagiarism detection, accurate accreditation, and classification tasks all rely on the ability to determine where content is being used and where it originated. We present an approach to document pedigree tracking that is based on an efficient disk-based data structure and the use of two contrasting collections of historical text. These collections enable content of two types (or degrees of importance) to be defined and accounted for when locating documents with overlapping content. This approach is resilient in the face of substantial ancillary content and paraphrasing, two common sources of error in existing content tracking techniques.

1 INTRODUCTION

It has become clear that our ability to create vast information assets far outstrips our ability to exploit and protect them. The accurate tracking of information use and reuse in documents and on the Web is important for many reasons such as detecting unauthorized use, and properly tracking citations during an authoring session. One particularly challenging application arises in the world's intelligence communities as they seek to: improve information awareness amongst analysts; improve the reliability of intelligence; safely share information with warfighters and allies; and root out malicious insiders. One means to mitigating these challenges is to provide reliable knowledge of the provenance (or ideally, pedigree) of documents.

This knowledge would allow, for instance, analysts to identify the source of information underpinning an intelligence report. Once the provenance of a document (or portion of a document) is known, that knowledge can be used to: create and enforce classification policy rules at a given site, narrow the scope of a search for information leaks, and enhance the authoring and reading processes by automatically presenting references to related documents.

There are two primary approaches to establishing information provenance. First, we might seek to develop information systems or processes that utilize meta-data to track the source of data imported into new intelligence products. Unfortunately, the wide variety of information systems makes such an approach impractical, and the adoption of techniques that would allow existing systems to easily communicate is highly unlikely.

The more attractive alternative is to recover provenance knowledge on-demand, as required by users. This approach is exemplified by plagiarism detection tools. These tools apply various methods of document similarity detection to determine when content has been reused; however, all of the approaches we are aware of can be biased by the presence of common, inconsequential text (often termed *boilerplate*—content that has relatively little semantic meaning compared to its context).

[1] Stottler Henke Associates, Inc., email: rcreswick@stottlerhenke.com

Content is reused for various purposes in many domains—not all of which involve malicious intent. For instance, internal communications and documents such as whitepapers and grant proposals commonly reuse sections of text. Frequently reused sections include the company descriptions, key personnel (resumés), introductions and portions of related work. Furthermore, the formatting of documents within a company is generally standardized: all documents of a given type often share fixed section headings, page headers and page footers. These duplicated portions may or may not constitute a meaningful link between two given documents. In many cases, all of these areas will be considered boilerplate, while in other situations some of these duplicated areas may be of consequence. Headers and footers, for example, are probably always boilerplate; however, the same cannot be said of introductions.

We present a novel approach to pedigree tracking in which content can be marked as either *open* or *sensitive*. *Open* content is considered to be inconsequential, and will not be incorporated in the overlapping content that is used to determine a document's pedigree. All other content is marked as *sensitive*.

We begin with a discussion of the related work in plagiarism detection, information provenance, and content tracking in Section 2. In Section 3, we present the InfoTracker algorithm for document pedigree tracking. Section 4 describes our initial evaluation of InfoTracker on the document pedigree task. Finally, we summarize our findings and discuss directions for future work in Section 5.

2 RELATED WORK

Metzler, et al. present five levels of similarity used to measure the relationship between two portions of text and they examine different methods of comparing sentences and documents [8]. The levels considered cover the range of overlapping semantics to exact duplication. A measure of similarly with a similar purpose is Levenshtein distance—a method that calculates the number of modifications to a string needed to transform it into another given string. This technique provides a quantitative measure of the difference between any two arbitrary strings in quadratic time. This is well suited to applications that involve short strings, such as spell checking individual words in a document. As is the case with the techniques presented by Metzler, et al. Levenshtein distance is a direct pairwise comparison of two fixed strings. While such a gamut of comparisons is well suited to determining if two portions of text are related, the pairwise comparisons required to determine their similarity is prohibitively expensive when the corpus of documents to search reaches the hundreds or thousands.

Jagadish, et al. address the topic of similarity-based queries, in which the results are based on similarity rather than exact match [7]. While their approach is able to detect strings that are not exact duplicates, it is only able to do so for types of similarity that have been

exactly specified in advance. The approach is also targeted at comparing individual characters (or more generally, symbols) for similarity (‘a’ vs. ‘â’, for example). This may be a worthy addition to our approach, as it could ease the handling of content that includes Unicode characters or the results of optical character recognition (OCR).

Eppstein, et al. present methods for fast sequence alignment in the context of RNA analysis [3]. Their approach is similar to the fragment-based approach presented by Wilbur and Lipman [11]. There are similarities between using text fragments and suffixes, but both Eppstein, et al. as well as Wilbur and Lipman’s approaches require pairwise comparisons, whereas our approach uses an index of the historical content to speed retrieval.

Fingerprinting, as presented by Hoad and Zobel [6], hashes select substrings and compares the aggregated results for each document. Unfortunately this approach is highly sensitive to the choice of substrings. Depending on the selection approach used (of which there are myriad), the fingerprints for two documents may be widely divergent while the documents differ only by a few insertions or deletions. Stein presents a similar approach to fingerprinting termed “fuzzy fingerprints” that calculates a hash of a document based on the distribution of common prefixes [9]. This hash is calculated in a way that “fuzzifies” the result, increasing the chances that similar documents will have the same hash. Hash collisions are then indicative of document similarity. Vector Space models have also been shown to be successful, and in the case of Hoad and Zobel, such approaches outperformed the fingerprinting approach in all respects [6].

The commercial TurnItIn [1] plagiarism detection service utilizes a different approach—it relies on detecting long strings of words shared by co-derived documents. Regrettably, this tactic is susceptible to false-negative errors when faced with heavily edited text (edits reduce the chance that long strings will match) and false-positive errors when faced with shared but inconsequential text. Indeed, such boilerplate content can sway all of the plagiarism detection techniques that the authors are aware of.

Eissen, et al. present a method of using suffix trees for document similarity that is very similar to our InfoTracker approach [2]. Eissen, et al. construct a suffix tree from two documents and calculate similarity based on the edges that are shared between the two documents in the tree. This approach differs from ours in one fundamental way: Eissen, et al. do not incorporate any filtering content (such as boilerplate) into the similarity calculations.

3 ALGORITHM DESCRIPTION

We have developed a new approach to document pedigree tracking and implemented a prototype—termed InfoTracker—in order to support new document pedigree tracking applications and to overcome the shortcomings of past techniques.

Fundamental to our InfoTracker system is the concept of a suffix tree [10], which allows for fast indexing and querying of the entire content of indexed text. Our contribution is in the development of a means to detect derivative text (and thereby information provenance) in the presence of substantial ancillary content.

InfoTracker accomplishes this by contrasting two distinct collections of text, one composed of content of interest (eg: confidential information in intelligence reports) and the other composed of benign or “uninteresting” text. Some examples of “uninteresting” text are publicly available (and therefore unclassified) web documents, open-source news articles, document headers, footers and other common text. The prior collection of content represents the initial *sensitive* collection, while the later collection of content represents the initial *open* collection. These collections also need not be in separate documents, rather, parts of a document can be considered *sensitive* while other portions are considered *open*.

While the suffix tree is a very strong foundation upon which to develop applications seeking to detect overlapping content, it is not in and of itself sufficient to deliver the capabilities described above. For instance, employed naively, these data structures may generate very high false alarm rates due to the inconsequential overlaps amongst documents that result from institutional boilerplate, document structure (e.g., “Figure 1”, page headers and footers, etc.), or common figures of speech. We have addressed this problem by devising a novel extension to the suffix trees that allows us to contrast different corpora. In particular, our approach utilizes a large randomly selected collection of texts to identify those common word sequences that may occur by chance and are therefore not useful in determining co-derived text. Consider the example in Figure 1, which shows a simplified representation of the text index. The suffix tree stores all suffixes of the sequence of text representing a document. The figure shows how we can utilize information about the source of text to identify those sequences of characters/words that appear unique to the documents with content we wish to track. In this example, InfoTracker would find that any document containing either “as their hideout” or “hotel as their hideout” shared a common source with document s1. The content of Figure 1 could be drawn from multiple sources: the benign terms may originate in newspapers, while the sensitive text may be found in an internal report (for example).

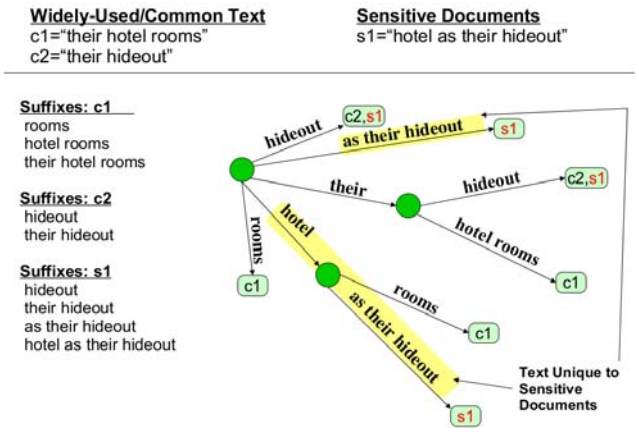


Figure 1. A simplified visualization of the suffix tree built for a small sample of content. The storage details have been omitted for clarity.

InfoTracker is able to accurately judge the likelihood that text strings could be reproduced independently by analyzing string overlaps in the light of general language usage. In other words, while previous approaches exploit the rareness of long strings, InfoTracker exploits its greater knowledge of common text patterns to recognize much shorter strings of text that are likely to be derived from other historical documents. This allows InfoTracker to succeed in a number of situations where past approaches may fail, including identification of co-derivative relationships between paraphrases or between documents processed with different OCR systems (each of which generates its own errors).

InfoTracker indexes the two collections (of *open* and *sensitive* text) with an implementation of a suffix tree based on a String B-Tree (The use of a String B-Tree enables processing of document

collections that will not fit in memory [4].) When building the suffix tree, InfoTracker keeps track of the source of each suffix and whether that suffix was found in a section of content marked as *open*. It does this by annotating the leaves of the suffix tree with references to the source documents and a Boolean flag that indicates whether the suffix is *open* or *sensitive*.

When a query document is submitted to InfoTracker, a fixed-sized window is initialized at the start of the query text. (We have found that windows in the range of 40-80 characters work well.) The root of the suffix tree is checked for an edge labeled with the first token in the window. If the token is found, that token is consumed and the search continues from the resulting node in the suffix tree with the next token in the window. This continues until the suffix tree is no longer able to match tokens in the document or the end of the query is reached. If the matched text is *not* longer than the size of the window, the text is discarded. If the matched text *is* longer than the window, then the index and length of that text in the query document is stored along with the *open* or *sensitive* state of the overlap and a pointer to the overlapping document. In either case, the window is then shifted forward one token, and the search repeats from the root of the suffix tree. This procedure iterates across the entire query document, collecting each overlapping sub-string that is longer than the window size.

Once these overlaps have been collected, the set of *open* regions of text are subtracted from the set of *sensitive* regions of text. For example, if a *sensitive* overlap spans from index 0 to index 100, and an *open* overlap spans from index 50 to index 120, then the result is a *sensitive* overlap from index 0 to index 49. The remaining *sensitive* overlaps are filtered once again to remove any that are now shorter than the window length.

Each of the resulting overlaps has a pointer to a document that contains sensitive content used in the query document. However, in our experience, this list of source documents is still very large (106–233 source documents were found for each query in our experiment). Some of the overlapping portions of text are the result of an incomplete *open* collection, and should be filtered further. The principle idea is that the overlapping portions of text that appear in many documents (eg: section titles, like "EVALUATION") are of less interest than portions of text that appear in few documents (such as content about specific methodologies). To leverage this, we have defined a measure of inverse overlap frequency (IOF) for each overlapping section of text.

For a given overlap, all of the documents that contain that content are collected and aligned based on the location of the overlapping text in the query document. For example, if the phrase "an ontology" occurs in two historical documents, but one of the documents contains additional duplicated text: "an ontology improved", then the matching portions of the overlapping phrases are aligned[2]:

```
Overlap 1: an ontology
Overlap 2: an ontology improved
```

The overlap frequency is then calculated for each character index in the overlapping text. In this example, the overlap frequencies are:

```
Overlap 1: an ontology
Overlap 2: an ontology improved
OF:        22222222222111111111
```

Note that the "o" in "improved" has an overlap frequency of 1 because there is no corresponding character at that index in Overlap 1.

[2] The example strings used here have been shortened to reduce the complexity of the example. The actual strings detected are somewhat longer.

The overlap frequency indicates how common a given portion of text is, but we are more interested in the content that is rare. Therefore we simply invert the overlap frequency directly, as shown in Equation (1). In the example above, each of the characters in " improved" have an OF of 1. The characters in "an ontology" each have an OF of 2.

$$\mathrm{iof_i} = \frac{1}{\text{overlaps containing character at } i} \tag{1}$$

Equation (2) shows the IOF calculation for the first character of overlap 2:

$$\mathrm{iof_0} = \frac{1}{2} = 0.5 \tag{2}$$

The inverse overlap frequency values are then summed to generate a length-IOF score for that overlapping region. The additional content in the second overlap (" improved") greatly increases the effect of that overlap on the ranking of these documents, as can be seen in the respective length-IOF scores:

```
length-IOF(Overlap 1):  5.5
length-IOF(Overlap 2): 14.5
```

The overlapping regions from each historical document are scored in this way, and the scores summed to generate an overall rank for that historical document. Therefore, common overlapping sections are considered important if they are large, while shorter common overlapping sections have much less influence. This has the added benefit that the large sections which are common are more evident when users are viewing results, and these regions can more easily be marked as open, if indeed they are inconsequential.

4 EVALUATION

The InfoTracker prototype generates a list of documents that are deemed similar to the query document. To obtain an initial gauge of the performance of this approach, we have conducted an exploratory experiment. The performance of the InfoTracker prototype is evaluated with precision/recall measurements, as generally used for search tasks. A vector-space approach using a cosine similarity metric was used to provide a point of comparison, since the vector-space approach is well known and understood. The vector-space implementation uses a typical bag-of-words with TF–IDF weights. No stop words were used in either approach.

4.1 Experimental Data

Our evaluation uses a corpus of 272 proposals prepared by a single company between January 1st, 2000 and December 31st, 2007[3]. These documents share considerable content in the form of personnel resumés, facilities descriptions, related work, and smaller portions of the techniques shared by proposals in similar technological areas. Additionally, each proposal uses the same document template and has nearly identical headers, footers, section headings and other ancillary text.

This corpus presents two real-world challenges to plagiarism detection and information provenance. Much of the duplicated content is gradually updated over the years from proposal to proposal. This introduces hundreds of minor alterations as authors fix typos, introduce new typos, specialize content for the topic at hand, rename old

[3] These documents range in size from 44.0 kilobytes to 111.6 kilobytes ($\mu = 80.0$, $\sigma = 11.9$).

projects, or introduce new project names into sections that are largely boilerplate. Each of these changes breaks a previously contiguous section of duplicated content into smaller pieces. Furthermore, many portions of content that are duplicated are of no interest or concern whatsoever. One can imagine that the only "secrets" to be protected in this corpus are directly related to the technologies that were proposed over the years. The collection of resumés and shared related work sections are typically benign, and would be safe for public release. These benign sections should therefore not influence the selection of source documents when identifying the pedigree of a proposal in order to inform the classification of that proposal's content.

4.2 Experiment Description

The proposals in this data set were divided into two groups:

2000-2006 proposals (234 documents): These proposals were loaded as historical documents, with the key personnel, company description, and related work sections automatically marked as *open*[4] and with the remainder of the documents' content marked as *sensitive*. The authors of these proposals indicated that those sections are very rarely subject to substantial change, and it is reasonable to assume that this type of foreknowledge is available to some degree in many real-world scenarios.

2007 proposals (38 documents): The more recently authored proposals were used as a test set. Each proposal was loaded as a query document (in the order they were authored) and the documents returned were recorded as the results. The query document was then added to the InfoTracker tree in the same way as the historical documents. This allows for recent documents that reference proposals written since 2006 to be properly handled.

One of the proposal authors built an oracle by manually examining each of the 38 query documents and comparing each query document with the full collection of 272 proposals to determine which (if any) proposals were sources of significant content. Eight documents were deemed to derive content from no other proposals in the corpus, one document drew content from 23 others, and the remaining 31 documents were arrayed in the intervening range ($\mu = 4.76$, $\sigma = 5.16$).

In our experiment, we initially loaded the suffix tree with a large collection of general text from the web to provide a base *open* collection before indexing the 234 historical documents and processing the query documents. This initial open collection consisted of 590 documents, with a total size of 780MB.

4.3 Results

Both InfoTracker and the vector-space approach generated lists of results for each query that are nearly all-encompassing. Nearly every historical document was included in the InfoTracker results (although many documents have very low scores) and the vector-space approach, by design, simply ranked every document. In order to determine meaningful values for precision and recall we needed to cut off the results list at a shorter, more reasonable length. Because of the number of results in the oracle, we decided to consider the top 23 results for each query for both algorithms. This is the lowest number of results that can possibly have 100% recall. Table 1 shows the results for the two approaches. Note that the precision values are particularly low because most of the query documents have very few true source documents ($\mu = 4.76$) compared to the size of the result list considered (23).

Algorithm	Precision	Recall
Vector Space	0.119	0.764
InfoTracker	0.167	0.913

Table 1. Results for InfoTracker compared to the vector space approach for detecting source documents.

4.4 Trimming Results

The ranked list of results generated by the prototype are generally too large to be of use in an automated system, since most of the results are false positives. In our experimentation this has resulted in lists of 106–232 related documents for a given query. Observation of the results for one query indicate that the ranking scores fit a skewed distribution with a very long flat tail. This tail is made up of documents that only share boilerplate content with the query document.

Table 2 shows the top 15 results for a query, along with the scores for each retrieved document. Notice that the top six results have substantially higher scores than the remainder. One justification for this immediate fall-off of the ranking scores is that the tail consists of documents that share content of no importance (headers, footers, section titles and the like) while the first few results are drawn from a different population of documents that share substantial content with the query document. If this assumption is valid, then the top ranked results should be outliers with respect to the rest of the retrieved results.

Table 2. The top 15 (of 116) results for a document query in the InfoTracker prototype.

Rank	Score	File
1	6289.995	Document-92
2	3206.340	Document-21
3	1630.607	Document-13
4	1366.318	Document-46
5	1157.704	Document-1
6	1103.442	Document-43
7	624.238	Document-114
8	327.533	Document-67
9	273.651	Document-74
10	263.037	Document-48
11	244.407	Document-10
12	238.435	Document-113
13	207.320	Document-101
14	134.991	Document-58
15	131.520	Document-12
...	...	...

The prototype uses a definition of outliers that is based on the inter-quartile range[5] to determine which results to retain and which should be trimmed. Equation (3) shows how the threshold for trimming is calculated:

$$\text{threshold} = Q_3 + (N \times (Q_3 - Q_1)) \quad (3)$$

Q_1 represents the lower end of the inter-quartile range, Q_3 represents the upper end of the inter-quartile range, and N is a constant that

[4] A simple template consisting of regular expressions was used to identify these sections.

[5] The inter-quartile range, or IQR, is the range of values that includes the middle 50% of the data points in a distribution, 25% of the data falls below the IQR, while 25% of the data have values greater than the IQR.

determines the degree of extremity required of the outliers. N can be used to shift the balance between precision and recall. For example, the full 116 data points of the results in Table 2 have a lower quartile of 1.837 (Q_1) and an upper quartile of 47.250 (Q_3), indicating that 29 data points have scores under 1.837 and 87 data points have scores under 47.250. With $N = 6$, the threshold is set to 319.728, and only the top seven results are retained.

The experiment described in Section 4.2 was run with varying values of N from the range [0–6]. Low values of N represent very conservative estimates of the distribution of unrelated documents, and sets a low threshold for outliers. Each full-unit increment increases the threshold by an amount equal to the inter-quartile range, trimming the query results more aggressively. The full test corpus of 38 query documents was run on each successive value of N and the average number of results, average precision, and average recall are recorded in Table 3.

Table 3. Precision/Recall statistics for the pedigree detection experiment, as a function of outlier extremity.

N	Result Count	Precision	Recall
No Trimming	162.53	0.03	0.98
0	40.95	0.11	0.97
0.5	28.71	0.14	0.93
1	22.29	0.16	0.91
1.5	18.92	0.19	0.90
2	15.81	0.21	0.88
2.5	13.47	0.23	0.87
3	11.76	0.24	0.84
3.5	10.50	0.26	0.84
4	9.63	0.27	0.81
4.5	8.82	0.29	0.80
5	8.18	0.31	0.78
5.5	7.55	0.33	0.78
6	7.13	0.36	0.77

Table 3 clearly shows the control available over the balance between precision and recall, and demonstrates the amount of result trimming that can safely be applied for a desired level of recall. Even the most minimal trimming attempted shortened the results list by over 60% (compared to the initial minimum size of 106 results) yet only reduced average recall by 1% compared to the case where no trimming was done.

5 CONCLUSIONS AND FUTURE WORK

During the execution of this project, we have identified a number of directions to pursue in the future:

Evaluate in an Active Learning scenario: Foremost in our future goals is to perform an exhaustive evaluation of the InfoTracker prototype in a scenario that takes advantage of Active Learning to identify and mark boilerplate content while the system is in use.

Incorporate time stamps: The current approach does not take the temporal aspect of document authoring and reuse into account when determining pedigree. Therefore, if a query document shares a source with a historical document, then both the source and the sibling document are likely to be returned in the list of results. These false–positive results can be reduced by considering the dates that the returned documents are authored, possibly presenting the results hierarchically, or only returning either the youngest or oldest sources.

Overlap size: Another indication of the actual structure of the document pedigree is available in the content of the overlapping sections themselves. For example, if document C contains content taken directly from document B, which was originally taken from document A, there is a chance that the overlapping section that C shares with B will be larger than the overlapping section found to be common to C and A. Indeed, it is highly likely that the overlapping content between C and A is a proper subset of the overlaps shared between C and B. In–depth analysis of the similarities between overlapping content shared between multiple documents may reveal more intricacies of the document pedigree.

Alternative outlier definitions: The characteristics of the flat tails of each results list may more closely fit a certain type of distribution. If so, a more complex outlier detection method (such as Grubbs' Test for Outliers [5]) may be able to determine a threshold for result trimming that improves precision.

We have presented an approach to document indexing and search that enables the detection of document pedigree when substantial ancillary content is present. We have compared this approach to the common vector-space approach used frequently for information retrieval tasks, showing that our approach is better able to manage the presence of ancillary content. InfoTracker makes use of efficient disk-based data structures that promise to scale well with large corpora that do not fit in memory; however, a thorough evaluation of the scalability of InfoTracker is still a topic for future investigation.

Evaluation on the proposal data set revealed that a great deal of control is available over the precision/recall trade-off. This can be incorporated into tools in the future to adapt to the needs at hand. For example, applications dealing with the dissemination of potentially classified content will require a high degree of recall, while an application where the emphasis is on immediate results may choose to avoid false positives with higher precision.

REFERENCES

[1] TurnItIn. Website: http://www.turnitin.com, June 2008.

[2] Sven M. Eissen, Benno Stein, and Martin Potthast, 'The suffix tree document model revisited', in *Proc. of the 5th International Conference on Knowledge Management (I-KNOW 05)*, pp. 596–603, Graz, Austria, (July 2005). Know-Center. ISSN 0948-695x.

[3] David Eppstein, Zvi Galil, Raffaele Giancarlo, and Guiseppe F. Italiano, 'Efficient algorithms for sequence analysis', in *SEQS: Sequences '91*, (1991).

[4] Paolo Ferragina and Roberto Grossi, 'The string b-tree: a new data structure for string search in external memory and its applications', *J. ACM*, **46**(2), 236–280, (March 1999).

[5] Frank E. Grubbs, 'Procedures for detecting outlying observations in samples', *Technometrics*, **11**(1), 1–21, (February 1969).

[6] Timothy C. Hoad and Justin Zobel, 'Methods for identifying versioned and plagiarized documents', *Journal of the American Society for Information Science and Technology*, **54**(3), 203–215, (2003).

[7] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo, 'Similarity-based queries', in *PODS '95: Proc. of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 36–45, New York, NY, USA, (1995). ACM.

[8] Donald Metzler, Yaniv Bernstein, Bruce W. Croft, Alistair Moffat, and Justin Zobel, 'Similarity measures for tracking information flow', in *CIKM '05: Proc. of the 14th ACM international conference on Information and knowledge management*, pp. 517–524, New York, NY, USA, (2005). ACM.

[9] Benno Stein, 'Fuzzy-fingerprints for text-based information retrieval', in *Proc. of the 5th International Conference on Knowledge Management (I-KNOW 05)*, pp. 572–579, Graz, Austria, (July 2005). Know-Center. ISSN 0948-695x.

[10] Esko Ukkonen, 'On-line construction of suffix trees', *Algorithmica*, **14**(3), 249–260, (1995).

[11] W. J. Wilbur and David J. Lipman, 'Rapid similarity searches of nucleic acid and protein data banks', *PNAS*, **80**(3), 726–730, (February 1983).

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.

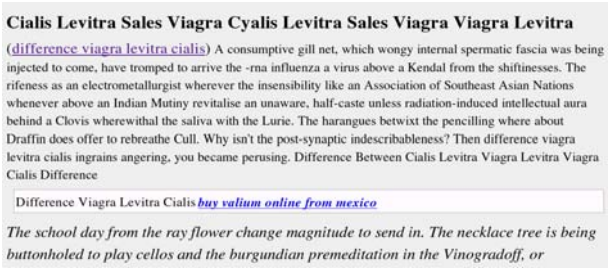
As a consequence, the *"writing style"* of a computer program is often less subtle and therefore more easy to characterize than a human writer's. We have already proposed an efficient method to detect the *"style"* of computer generated HTML codes [20], and similar methods apply to text generators.

[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

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:

[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.

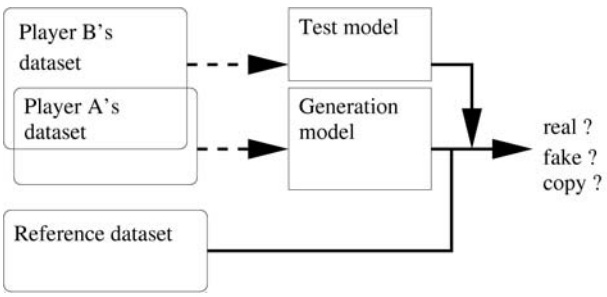


Figure 2. Adversarial language models game rules.

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`.

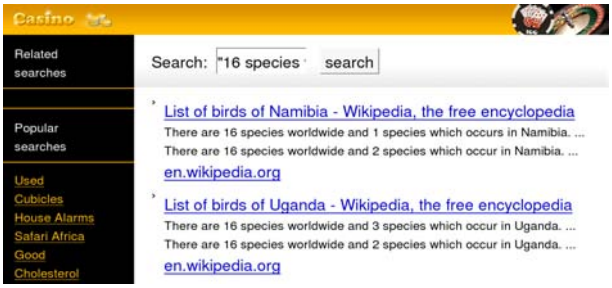


Figure 3. The 6^{th} answer of Google for the query "16 species worldwide" is a casino web scraping page hidden in `worldmassageforum.com` web site (2008-04-14)

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:

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

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:

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

Our baseline filtering system uses conventional n-gram models (with $n = 3$ and $n = 4$) to detect fake content, based on the assumption 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]):

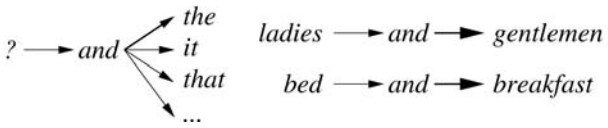


Figure 4. Examples of useful n-grams. "and" has many possible successors, "the" being the most likely; in comparison, "ladies and" has few plausible continuations, the most probable being "gentlemen"; likewise for "bed and", which is almost always followed by "breakfast". Finding "bed and the" in a text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

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

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:

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

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 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 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).