Intrinsic Plagiarism Detection using Complexity Analysis

Leanne Seaward and Stan Matwin
University of Ottawa
2096 Madrid Avenue, Ottawa, ON,K2J 0K4
leanne_seaward@yahoo.ca, stan@site.uottawa.ca

Abstract: We introduce Kolmogorov Complexity measures as a way of extracting structural information from texts for Intrinsic Plagiarism Detection. Kolmogorov complexity measures have been used as features in a variety of machine learning tasks including image recognition, radar signal classification, EEG classification, DNA analysis, speech recognition and some text classification tasks (Chi and Kong, 1998; Zhang, Hu, and Jin, 2003; Bhattacharya, 2000; Menconi, Benci, and Buiatti, 2008; Frank, Chui, and Witten, 2000; Dalkilic et al., 2006; Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008). Intrinsic Plagiarism detection uses no external corpus for document comparison and thus plagiarism must be detected solely on the basis of style shifts within the text to be analyzed. Given the small amount of text to be analyzed, feature extraction is of particular importance. We give a theoretical background as to why complexity measures are meaningful and we introduce some experimental results on the PAN'09 Intrinsic Plagiarism Corpus. We show complexity features based on the Lempel-Ziv compression algorithm slightly increase performance over features based on normalized counts. Furthermore we believe that more sophisticated compression algorithms which are suited to compressing the English language show great promise for feature extraction for various text classification problems.

Keywords: plagiarism, Kolmogorov, complexity, compression, machine learning

1 Introduction

Intrinsic plagiarism analysis involves analyzing a document for style changes which would suggest that certain passages have been written by a different author and are therefore plagiarized. It is closely related to authorship attribution and stylometry (Stamatatos, Fakotakis, and Kokkinakis, 2000; Stein and Meyer zu Eissen, 2007). Intrinsic plagiarism analysis is a very challenging problem because one has a small amount of text for global analysis and one must locally analyse very small portions or chunks of that text for style shifts. Authorship attribution normally uses several documents for author fingerprinting and tests possible authorship on an entire text document.

Because of the limited data available for this task and the difficulty of the problem, feature extraction is very important. Plagiarism analysis tools and authorship attribution models attempt to fingerprint an author's individual writing style using style features such as normalized counts of lexical and vocabulary richness features such as nouns, verbs, stop words, syllables per word etc (Stamatatos, Fakotakis, and Kokkinakis, 2000; Stein and Meyer zu Eissen, 2007). In addition one may analyze a document for topic or cohesion words. One may also use readability indexes to determine if the level of writing shifts (Stein and Meyer zu Eissen, 2007).

Features are extracted globally (for the entire document) and then locally (per sentence or paragraph chunk). With the exception of n-gram methods, the text is generally viewed as a bag-of-words and structure is ignored. We introduce a method of using compression to extract Kolmogorov complexity features which contain information about the structure of style features within the text. Extracting such features is scalable and complexity features can be used in state-of-the-art machine learning algorithms such as Support Vector Machines, Neural Networks and Bayesian Classifiers. The small text sample makes complexity analysis more difficult than for the authorship attribution problem. However, this method still shows promise and given the difficulty of the problem, a modest improvement is still important.

2 Introduction to Plagiarism Detection

There are two main types of plagiarism analysis - Intrinsic and Extrinsic. Extrinsic plagiarism analysis compares the document of interest to a corpus of reference documents (web pages, text books etc.) and tries to find passages which were copied from the reference collection. In contrast, intrinsic plagiarism detection uses no reference collection and tries to determine plagiarized passages by analyzing style changes within the document. Intrinsic plagiarism detection is closely related to author fingerprinting or stylometry.

Most research in plagiarism analysis focuses on extrinsic plagiarism analysis. If one assumes that the reference collection is complete, then extrinsic plagiarism analysis is a somewhat easier problem due to the fact that one must simply find the match between the plagiarized passage and the corresponding passage in the reference collection. The difficulty lies in reducing the computation time and detecting obfuscation attempts.

Obtaining a reference collection of all possible sources of plagiarism is impossible. Not all books are in electronic format and indexing all books for inclusion in such a corpus is a formidable task. There is always the possibility that a student has plagiarized from a document which is not available for indexing such as a paper from another student at another university.

One imagines that a robust plagiarism analysis tool would use both intrinsic and extrinsic plagiarism analysis. This is similar to the way a human expert such as a teacher or professor would analyze student papers for plagiarism. One may also use intrinsic plagiarism analysis to pre-select suspicious passages which can then be passed to an extrinsic plagiarism detector. It is always more desirable to have access to the plagiarized document as this removes all doubt as to the suspected plagiarism.

Intrinsic plagiarism analysis is related to authorship attribution and generally uses stylometry features which may consist of normalized counts of lexical features such as nouns and verbs as well as measures such as average sentence length and average word length. Intrinsic plagiarism detection may also use readability indexes and as measures which compute the divergence of the distribution of lexical elements to the expected probability distribution. With the exception of readability indexes, features are extracted as if each chunk in the text is a bag-of-words. Humans do not read or write Bags-of-words and so this approach is counterintuitive and loses information.

The need arises for a way of measuring structure of a text in a meaningful way which can be used as a feature in style analysis. It is also necessary that such a measure can be computed in an efficient and scalable manner.

The structure which we are measuring must be meaningful for the classification task at hand. We propose Kolmogorov Complexity measures as a way of measuring structural complexity of lexical elements in order to fingerprint author style.

3 Kolmogorov Complexity Measures

This paper introduces Kolmogorov complexity measures as style features in intrinsic plagiarism analysis. The basic idea is that each segment of text has a distribution with respect to a set of word classes. For example with respect to the word class noun – the text has a distribution of noun words and non-noun words. This can be though of as a binary string which has a 1 for each noun word and a 0 for each non-noun word. This binary string represents the distribution of noun words in the text.

For example, suppose we have the string: "Billy walked the dog yesterday." The nouns are "Billy" and "dog", the noun distribution is '10010'. Likewise the only verb is "walked" so the verb distribution is '01000'. Similarly if we look at short words (those with one syllable) vs. long words the distribution is '11001'. There is a different distribution for any possible class of word type.

In Figure 1 we see how a text can be decomposed into a representation for each word class. Once we have this decomposition we would then like to quantify the structure for use in a machine learning algorithms.

Two sentences may have the same ratio for a particular feature but the distribution could be different. Suppose two sentences have the following structure for short words vs. long words.

01000011110100001000111101000000 1
000000001111110000001110000001111

Both representations have the same number of long vs. short words (0 vs. 1) but the

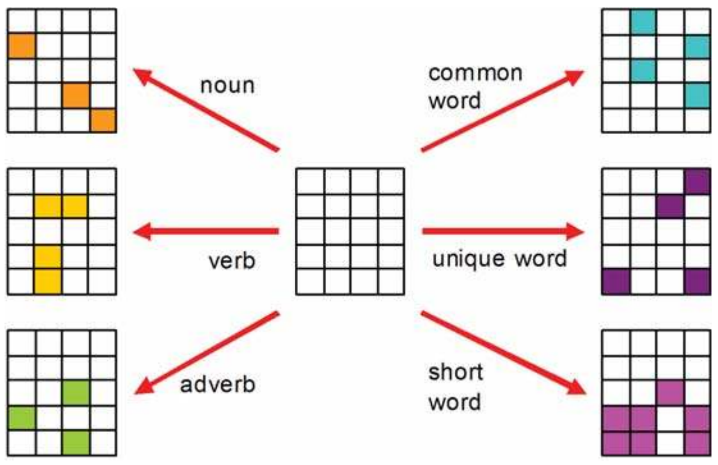

Figure 1: Decomposing a sentence into a variety of word class distributions for complexity analysis

first representation is more random and complex than the second. It is desirable to quantify this degree of randomness or complexity. One such method of doing so is Kolmogorov complexity measures.

4 Kolmogorov Complexity

Kolmogorov complexity, also known as algorithmic entropy, stochastic complexity, descriptive complexity, Kolmogorov-Chaitin complexity and program-size complexity, is used to describe the complexity or degree of randomness of a binary string. It was independently developed by Andrey N. Kolmogorov, Ray Solomonoff and Gregory Chaitin in the late 1960's (Li and Vitanyi, 1997).

In computer science, all objects can be viewed as binary strings. Thus we will refer to objects and strings interchangeably in this discussion. The Kolmogorov complexity of a binary string is the length of the shortest program which can output the string on a universal Turing machine and then stop (Li and Vitanyi, 1997).

It is impossible to compute the Kolmogorov complexity of a binary string. However there have been methods developed to approximate it. The Kolmogorov complexity of a string x, denoted as $K(x)$, can be approximated using any lossless compression algorithm (Li and Vitanyi, 1997). A compression algorithm is one which transforms a string A, to another shorter string, B. The associated decompression algorithm transforms B back into A or a string very close to A. A lossless compression algorithm is one in which the decompression algorithm exactly computes A from B and a lossy compression algorithm is one in which A can be approximated given B. When Kolmogorov Complexity, or $K(x)$, is approximated, this approximation corresponds to an upper-bound of $K(x)$ (Li and Vitanyi, 1997). Let C be any compression algorithm and let $C(x)$ be the results of compressing x using C. The approximate Kolmogorov complexity of x, using C as a compression algorithm, denoted $K_c(x)$, can be defined as follows:

$$K_c(x) = \frac{Length(C(x))}{Length(x)} + q$$

where q is the length in bits of the program which implements C. In practice, q is usually ignored as it is not useful in comparing complexity approximations and it varies according to which programming language implements C. If C was able to compress x a great deal then $K_c(x)$ is low and thus x has low complexity. Likewise if C could not compress x very much then $K_c(x)$ is high and x has high complexity.

5 Compression Algorithms and Kolmogorov Complexity Analysis

Kolmogorov complexity can be computed using any lossless compression algorithm. Once the text to be analyzed has been converted into a binary form related to a particular word class distribution, one simply applies a compression algorithm to determine the degree to which it is compressed. If it compresses a great deal then complexity is high and vice versa.

Our previous research has used generic compression algorithms such as run-length encoding and Lempel-Ziv (Zlib) compression. For this intrinsic plagiarism detection task, Zlib compression was used. It may be especially interesting to investigate compression algorithms which assume prior knowledge about the probabilities of lexical features or which are designed to maximize compression for language texts in a particular language.

Frank et al. (2000) investigated text categorization using statistical data compression techniques. They use a corpus of two classes of documents and train a statistical compression tool (prediction by partial matching or PPM) using each corpus. They attempt to classify documents by determining which compression model compresses it the most. They conclude that data compression techniques perform well but are inferior to state of the art machine learning techniques such as SVM or Neural Nets. No attempt was made to merge compression features with machine learning algorithms.

Thus we have three possibilities for compression algorithms:

1. An algorithm which assumes no prior knowledge and which can be used for any compression task, text or otherwise.
2. An algorithm which has some knowledge or prior probabilities and is trained for a specific compression task (such as compressing English text).
3. An algorithm which is specifically trained with respect to a corpus which corresponds to a class which we want to predict. This is very closely related to Kolmogorov Similarity Metrics.

The question arises as to whether all or any of these compression algorithms yield meaningful features and if so why. Compression analysis for machine learning has been done in a wide variety of fields. In fact much of the research has been done by those outside of machine learning who may or may not even know they are performing machine learning and who seem to have done little research into compression and complexity analysis and have no idea why their method works, only that it does.

Compression/complexity analysis has been used in many classification tasks such as image recognition, radar signal classification, EEG classification, DNA analysis, speech recognition and some text classification tasks (Chi and Kong, 1998; Zhang, Hu, and Jin, 2003; Bhattacharya, 2000; Menconi, Benci, and Buiatti, 2008; Frank, Chui, and Witten, 2000; Dalkilic et al., 2006; Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008).

The method proposed here is different then those which use Kolmogorov Complexity measures to compute the distance between the object to be classified and a corpus of training data. As this is intrinsic plagiarism analysis there is no set of documents for which we can find a similarity metric. We can only compare local text to the global document. We can use a statistical compression algorithm and this is somewhat related to similarity metrics but it is not the same. We are not explicitly using the concept that "like compresses with like". Moreover, such compression measures can be used in a variety of machine learning algorithms such as support vector machines, neural networks and decision trees. We can also use boosting and meta algorithms such as bagging and ADAboost. What we are doing is finding a measure for each different distribution as to how well its complexity can be described by the compression algorithm.

6 Using Compression to Estimate Complexity

Suppose we have a statistical compression model which has been trained on a variety of English text and we compress two text samples and find that one compresses much more than the other. This means that one text was much more alike to general English text than the other was.

Now suppose we extract the noun representation of both of those texts and compress them using a statistical compression algorithm which has been trained on noun representations of English text. The one which compresses the most is closer to the normal noun distributions of English text.

What if we use a compression algorithm that has no prior training such as Lempel-Ziv? Is it still meaningful? The answer is

Classifier	Complexity features	Plagiarism	Recall	Precision	F-measure
SVM	no	yes	0.651	0.538	0.589
	no	no	0.615	0.719	0.663
	yes	yes	0.671	0.521	0.587
	yes	no	0.617	0.752	0.678
Neural network	no	yes	0.619	0.510	0.559
	no	no	0.593	0.695	0.640
	yes	yes	0.670	0.548	0.603
	yes	no	0.626	0.737	0.677

Table 1: Results on using feature sets with and without complexity features with SVM and Neural Networks.

yes because we still have an idea of the complexity of the distribution of nouns. While it does not directly relate to the norms of the English language, it is still a meaningful measure of the complexity of that distribution. It relates the noun distribution to some distribution which can be most efficiently compressed by that compression algorithm (even if we do not know the distribution).

Research has shown this holds true (Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008). Dalkilic et al. (2006) have shown that Lempel-Ziv compression of text can be used to distinguish authentic text from non-authentic or computer generated text. They show that the compressibility of real texts is different than that of computer generated "nonsense" texts due to topic adherence. The idea is that when one writes a coherent text, ideas and words are repeated to increase readability.

With respect to text and compression, de Marcken theorizes that language learning is essentially a compression problem (De Marcken, 1996). If one has a great deal of knowledge about a language then one can build a model which maximizes the compressibility of text written in that language. Thus the compressibility of a text is a measure of how closely related the compression algorithm is to the text representation.

7 *Experimental Results*

The PAN 09 intrinsic plagiarism competition corpus consisted of 3091 annotated texts for training and 3091 texts for testing purposes (initially released unannotated). We extracted normalized counts and complexity counts for the following word classes:

Nouns	Stopwords
Verbs	Topic words
Pronouns	Common words
Adjectives	Passive words
Adverbs	Active words
Prepositions	Word length

Features were extracted locally and globally and the standard deviation amongst the local features was also computed. Zlib was used for compression.

A 50/50 training/test split was used on the training set to analyze the performance gained from adding complexity measures. The results were repeated for 10 random splits and averaged. Two classifiers were used – Support Vector Machine (SVM) and a Neural Network. Recall and precision are calculated per text chunk not per character (see Table 1).

For many classifiers tested such as regressions trees and support vector machines the F-measure performance gained by using complexity features was less than 2%. The neural network showed the most improvement with complexity measures as F-measure was increased 3.7-4.4%.

Previous classification tasks such as authorship attribution and spam filtering showed better results. The problem, as it was discovered, was the high degree of granularity required by the task. Complexity analysis does not do well with short text.

Using various feature selection tools it was found that complexity features and normalized count features were found in equal numbers in the highest ranked features. For example the top 10 features as determined by a Chi-squared feature evaluator is shown below.

As one can see in Table 2, 6 out of the 10 top ranked features are complexity features. This indicates that complexity fea-

Rank	Feature
1	Adjective complexity (l)
2	Adjective count (gsd)
3	Topic word complexity (g)
4	Verb word complexity (g)
5	Passive word complexity (g)
6	Active word complexity (g)
7	Preposition count (g)
8	Stop word count (gsd)
9	Avg. word length per sentence (gsd)
10	Topic word complexity (l)

Table 2: Top 10 ranked features for the intrinsic plagiarism task as calculated by a Chi-squared feature evaluator. l=local, g=global, gsd=global stantard deviation

tures are able to discriminate plagiarized vs. non-plagiarized passages as well as or better than normalized count features.

8 Conclusion

We introduce using compression to find features based on Kolmogorov complexity measures. We show why compression of text and word distributions results in meaningful features. Results in using complexity analysis in intrinsic plagiarism detection are promising. Performance is increased by a small amount and it seems as though complexity is not contributing to over fitting. More research needs to be done in using compression models which have prior knowledge of the language to be analyzed and/.or the prior probabilities of word classes. This would result in more meaningful complexity features which would likely aid in the difficult task of intrinsic plagiarism detection.

References

Bhattacharya, J. 2000. Complexity analysis of spontaneous EEG. *Acta Neurobiologiae Experimentalis*, 60(4):495–501.

Chi, Z. and J. Kong. 1998. Image content classification using a block Kolmogorov complexity measure. In *Proceedings of the Fourth International Conference on Signal Processing ICSP 1998*, Beijing, China.

Dalkilic, M. M., W. T. Clark, J. C. Costello, and Radivojac P. 2006. Compression to Identify Classes of Inauthentic Texts. In *Proceedings of the SIAM International Conference on Data Mining SDM 2006*, Bethesda, MD.

De Marcken, C. 1996. *Unsupervised language acquisition.* Phd thesis, Michigan Institute of Technology. http://www.demarcken.org/carl/papers/PhD.pdf.

Frank, E., C. Chui, and I. H. Witten. 2000. Text categorization using compression models. In *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209, Snowbird, USA. IEEE Computer Society Press.

Li, M. and P. Vitanyi. 1997. *An Introduction to Kolmogorov Complexity and its Applications.* Springer Verlag, Berlin, second edition.

Menconi, G., V. Benci, and M. Buiatti. 2008. Data compression and genomes: a two-dimensional life domain map. *Journal of Theoretical Biology*, 253(2):281–288.

Seaward, L., D. Inkpen, and A. Nayak. 2008. Using the Complexity of the Distribution of Lexical Elements as a Feature in Authorship Attribution. In *Proceedings of 6th International Conference on Language Resources and Evaluation LREC*, Marrakech, Morocco.

Seaward, L. and L. V. Saxton. 2007. Filtering spam using Kolmogorov complexity measures. In *Proceedings of the 2007 IEEE International Symposium on Data Mining and Information Retrieval (DMIR-07)*, Niagara Falls.

Stamatatos, E., N. Fakotakis, and G. Kokkinakis. 2000. Automatic Text Categorization in Terms of Genre and Author. *Computational Linguistics*, 26(4):461–485.

Stein, B. and Sven Meyer zu Eissen. 2007. Intrinsic plagiarism analysis with meta-learning. In *Proceedings of the SIGIR 2007 International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, PAN 2007*, Amsterdam, Netherlands.

Zhang, G., L. Hu, and W. Jin. 2003. Complexity feature extraction of radar emitter signals. In *Environmental Electromagnetics, 2003. Proceedings of the Asia-Pacific Conference on Environmental Electromagnetics CEEM 2003*, pages 495–498.