Using Compression to Identify Classes of Inauthentic Texts

Mehmet M. Dalkilic, Wyatt T. Clark, James C. Costello, Predrag Radivojac*

{dalkilic, wtclark, jccostel, predrag}@indiana.edu

School of Informatics, Indiana University

Bloomington, IN 47408

Abstract

Recent events have made it clear that some kinds of technical texts, generated by machine and essentially meaningless, can be confused with authentic, technical texts written by humans. We identify this as a potential problem, since no existing systems for, say the web, can or do discriminate on this basis. We believe that there are subtle, short- and long-range word or even string co-occurrences extant in human texts, but not in many classes of computer generated texts, that can be used to discriminate based on meaning. In this paper we employ the universal lossless source coding algorithms to generate features in a high-dimensional space and then apply support vector machines to discriminate between the classes of authentic and inauthentic texts. Compression profiles for the two kinds of text are distinct—the authentic texts being bounded by various classes of more compressible or less compressible texts that are computer generated. This in turn led to the high prediction accuracy of our models which support our conjecture that there exists a relationship between meaning and compressibility. Our results show that the learning algorithm based upon the compression profile outperformed standard term-frequency text categorization schemes on several non-trivial classes of inauthentic texts.

1 Introduction

When operating over a corpus of text there is a natural presumption that the text is meaningful. This presumption is so strong that neither the tools, like webpage search engines, nor the people who use them take into account whether, for example, a webpage conveys any meaning at all, even though the number of indexable webpages available is so large and growing [12]. And yet, a web search for the nonsensical sentence, "Colorless green ideas sleep furiously," yields scores of thousands of hits on Google, Yahoo, and MSN. Of course this is no ordinary sentence—it is Noam Chomsky's famous sentence that he constructed to illustrate that grammar alone cannot ensure meaning [24]. While the sentence is syntactically correct and can be parsed, it does not possess any real meaning. But the important point is that the sentence *is* meaningless and has become part of the searchable text of the web indistinguishable from any other sentence.

Single sentences can seldom convey enough meaning and are therefore combined into texts or documents to provide some larger, more complex information. According to linguists, texts exhibit not only sentential structure, but also higher levels of structure, for example, the so-called *expository structure*. Expository structures are meant to be *informative*, that is, scholarly, encyclopedic, and factual as opposed to, say, those intended for entertainment. These higher level distinctions can be somewhat problematic if taken too literally, but are useful nonetheless. We can take other perspectives too: there are global patterns that are *only* manifested when the text is examined in its entirety. For example, one kind of global text pattern is the adherence to a *topic* [10]. Another example is *discourse*—the different kinds of meaning derived solely from the arrangement of sentences.

We make several observations here. First, human creation and recognition of meaningless sentences is not very difficult. Second, human creation and recognition of meaningless texts is likely not much more difficult, though for a human reader, deciding whether a text is or is not from a set of both types will require more time than deciding whether a sentence is meaningful or not. Third, well-known and utilized web search tools currently do not distinguish between meaningless and meaningful texts.

To make clear the class of problem we are interested in examining, we provide the following definitions:

Definition 1.1. *An* inauthentic *text (or document) is a collection of several hundreds (or thousands) of syntactically correct sentences such that the text* as a whole *is not meaningful. A set of inauthentic texts will be denoted by $\mathcal{I}$ with possible sub- or superscripts.*

Definition 1.2. *An* authentic *text (or document) is a collection of several hundreds (or thousands) of syntac-*

*Corresponding author

tically correct sentences such that the text as a whole *is meaningful. A set of authentic texts will be denoted by* $\mathcal{A}$ *with possible sub- or superscripts.*

Now consider a scenario in which inauthentic texts are not human generated, but are automated and embellished further with figures, citations, and bibliographies. Without dedicated human scrutiny such texts can escape identification and easily become part of searchable texts of cyberspace. Such a scenario recently played out when an automated inauthentic text was accepted to a conference without formal review[1], although we are not aware of the mechanism that led to its acceptance. There are, in fact, scores of systems that generate inauthentic data and short texts, ranging from names, to email, and as demonstrated above, to text[2]. While no direct numbers exist, we believe that currently most of the text in cyberspace is authentic. There is no reason to believe, however, that this will hold true in the future. Indeed, it is not too hard to foresee schemes in which "information pollution" (certainly one perspective of inauthentic texts) is produced to confound search results, perhaps obscuring authentic texts to the point of uselessness–Google spamming is one such kind of scheme.

We can identify what we believe to be an interesting problem: *create a classifier that distinguishes between authentic and computer generated inauthentic texts given a set containing both.* Clearly, we cannot make direct use of syntax as eluded to above. Furthermore, while text mining is gaining a lot of attention, no formal components of any of these models seek to either identify or even distinguish meaningful texts from meaningless ones.

In perusing collections of inauthentic and authentic texts, we observed there seems to be a kind of information flow (or "semantic coherence" according to linguists) that the authentic texts possessed, but the inauthentic did not. Given that this flow is a kind of pattern, we arrived at the following conjecture:

CONJECTURE 1.1. *There is a discernable and actionable correlation between meaning and compression in texts.*

In Figure 1 we have given a stylized depiction of how we imagine compression and meaning are related with respect to expository texts. Given that Conjecture 1.1 holds, we might be able to somehow exploit text compressibility to separate the two kinds of text.

[1]http://pdos.csail.mit.edu/scigen/

[2]http://uzful.org/generators_online/on_line_generators.php

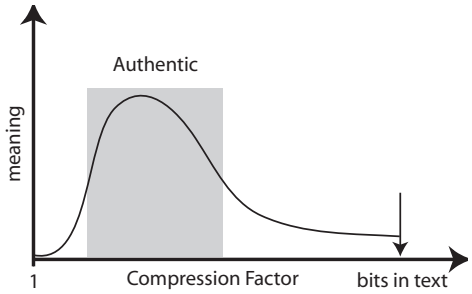

Figure 1: A stylized rendering of the relationship between compression and meaning in an expository text. Observe that the compression factor, the fraction of original document size to compressed size is between approximately one (book keeping makes the compressed size slightly larger) and the number of bits in the document. Our conjecture is that meaningful expository texts lie somewhere between these extremes and can be used to help distinguish between $\mathcal{I}$ and $\mathcal{A}$.

In this paper we use supervised learning to distinguish between authentic texts and several classes of inauthentic texts. Our method is based on employing the universal source coding algorithms to generate features in a high-dimensional space, then applying support vector machines to discriminate between the classes. Our results indicate this is indeed possible for a number of different kinds of inauthentic texts. Based on the compression profiles we observed, the authentic texts were bounded by various classes of more compressible or less compressible texts. First, these results support our conjecture that there *is* a relationship between meaning and compressibility. Second, there is a complex relationship between the objective measure of information (as defined by Hartley [14]) and the generally accepted notion of meaning. Third, results indicate that the learning algorithm based upon the compression profile outperforms standard text categorization schemes on several non-trivial classes of inauthentic texts. Last, on a deeper level, there seem to be subtle, long range patterns in meaningful texts that prove elusive for computer generated texts to emulate.

The remainder of this paper is organized as follows. In Section 2 we discuss approaches related to the problem of text categorization, the use of compression methods in text mining and latent semantic analysis, an emerging technique used to measure textual coherence. In Section 3 we describe methodology used in this study. We start with a coverage of the standard compression algorithms that we slightly modified for this study. We then proceed with data collection and representation procedures and describe our method for predictor construction and evaluation. Experiments

and results of our approaches are discussed in Section 5, while the concluding remarks are contained in Section 6.

2 Related Work

2.1 Text Categorization. Distinguishing between authentic and inauthentic technical documents is clearly a binary text categorization problem. In such a setting, one of the many text categorization schemes could be followed [34], but arguably the best performance has been achieved using word-based data representations, *e.g.* bag of words or TF-IDF [16], and subsequently employing supervised learning algorithms. It has been shown that even simple schemes such as naïve Bayes can provide satisfactory accuracy, while the support vector machines have gained reputation as the most appropriate model [16]. There are several reasons, however, why the bag-of-words data representation may fail to do well for this classification task, at least as the only model. First, it is insensitive to the order of words, and it can be easily argued that a random permutation of words or sentences within an authentic document can make such documents indistinguishable. Second, standard text categorization models are inherently category-specific and frequently rely on the identification of just several words, possibly by feature selection filters, to be successful. On the contrary, authentic papers can generally belong to any category of documents and are defined by semantics, not necessarily by the word frequencies. Third, documents for which a large corpus cannot be easily collected may not be accurately predicted due to the small dataset sizes. For such categories, it would be important to use "semantic models" from other authentic papers that could help in the categorization task. Thus, some sort of context sensitive analysis must be invoked.

2.2 Compression Methods in Text Mining. Compression methods have already found respectable application in various areas of text mining, as recently predicted by Witten *et al.* [33]. Some of the areas include extraction of generic entities, token segmentation, acronym extraction [32] and text categorization [32, 10]. Other attractive applications include authorship attribution [20] or even plagiarism detection [18]. In this section we review some of the techniques most relevant to our work.

Compression methods represent a viable approach in cases when the context needs to be incorporated into the classification scheme. In essence, lossless compression is capable of finding frequently occurring patterns in text based on a single (or multiple passes) through the document. These approaches are generally trained on a corpus of related documents and can produce a code that can later be exploited to detect other related documents. This approach has been adopted by Frank *et al.* [10] who attempted to classify text into various categories based on the partial string matching compression algorithm [8], called PPM (Prediction by Partial string Matching). PPM is a variant of arithmetic coding, introduced by Pasco [26] with adaptively changing symbol/pattern probabilities. In particular, Frank *et al.* train a compression model on each of the categories and when presented with a new document, attempt to compress it with each of the trained models. This approach was novel and was shown to perform surprisingly well on pairs of categories. However, it proved not to be competitive with standard text categorization techniques using bag-of-words representation and support vector machines [10, 32]. The major reason for such a performance were very subtle differences between some document classes. For example, on the Reuters-21578 collection, it proved to be hard to distinguish between *grain* versus *wheat* or *corn* versus *grain* categories. These articles predominantly use the same terminology, but were classified differently based on just a few important words. Clearly, such key words can be easily detected by a naive Bayes model or a support vector machine and are indistinguishable to a compression system.

Kukushkina *et al.* [20] applied compression algorithms to authorship attribution. A corpora that consisted of the text documents written by 82 different authors was compressed using 16 compression packages, *e.g.* arj, compress, gzip, ppm, rar. For two documents d_1 and d_2, they defined relative complexity $C(d_1|d_2)$ as $C(d_1|d_2) = |compress(d_1)| - |compress(d_2 d_1)|$, where $|\cdot|$ indicates the number of bits used to encode a document, $d_2 d_1$ are two concatenated documents, and *compress* is an arbitrary compression algorithm. Although it is not clear how the variability in lengths between different authors was accounted for, the performance results showed that half of the used compression algorithms were able to correctly identify authors in more than 50% of the cases.

Similarly to the approach by Frank *et al.* [10], Khmelev and Teahan [18] defined a repetition-based measure that could be used for text categorization and plagiarism detection. In particular, an R-measure was defined as the "normalized sum of the lengths of all suffixes of the text repeated in other documents in the collection." The authors showed that the idea was promising and reached interesting conclusions about plagiarized articles based on the Reuters Corpus Volume 1.

Other applications of compression algorithms include identification of structured sources, where a struc-

tured source outputs specially formatted sequences, or even acronym finders where acronyms were found as the occurrences of typically capital letters with significantly different letter or language statistics [32, 33].

2.3 Latent Semantic Analysis. Latent Semantic Analysis (LSA) is a popular computational approach to discerning the meaning of words as they appear in a particular context. Because LSA is so ubiquitous and seems to share, at least in part, an aim of establishing meaning, we briefly discuss the related work on LSA.

LSA relies upon singular value decomposition [5] as a basis for finding semantic relatedness of words from the context of text passages. It leverages the presence of words occurring in specific context; therefore, a context becomes the grouping of terms within that context, while the terms can be described as a relation to the context. This grouping of words to a context is very powerful, because relationships can be found between words that would be lost by simple co-reference [2] or using dictionary word definitions [22].

One perspective that seems in line with our work is to consider LSA as a means of measuring the coherence (or flow) of a text. From this perspective, it has been used to test reading comprehension [21] and improve the text searching [9, 11, 15].

3 Methods

3.1 Lempel-Ziv Compression Algorithms. The data representation used here is based on the original version of the Lempel-Ziv source coding algorithm [35] and its extension proposed by Bender and Wolf [4], hereafter referred to as the Bender-Wolf algorithm. In order to make this paper self-contained, we briefly explain these algorithms below.

Consider a sequence of n symbols $s_{[1,n]} = (s_1, s_2, s_3, ..., s_n)$ such that $s_i \in A$, where $i \in \{1, 2, \ldots n\}$ and A is an alphabet of fixed size. The goal of the Lempel-Ziv algorithm is to encode sequence $s_{[1,n]}$ in one pass so as to minimize the number of bits used to represent the output sequence. At each explored position i along the sequence, it searches for the longest prefix of $s_{[i,n]}$ such that the matching sequence starts within the previous w symbols $s_{[i-w,i-1]}$ called the *window*. If a match is found, it is encoded as $\langle 1, i-j, m\rangle$, where j is the starting position of the matching subsequence and m is the length of the longest match. Clearly, $i > j \geq i-w$. In the case a match is not found, it is encoded by $\langle 0, s_i\rangle$. The one-bit flag at the beginning of each codeword indicates the presence (1) or absence (0) of a match and determines the way the rest of the codeword is interpreted. In the former case, both the match position $(i-j)$ and match length (m) are encoded using $\lceil \log_2 w \rceil$ bits, while in the latter case the symbol s_i is encoded using 8 bits, assuming the ASCII representation. After each step, current poisition i is incremented either by 1, if a match is not found, or by m.

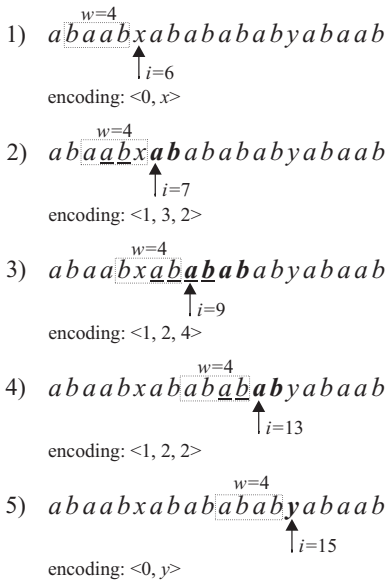

Figure 2: An example of the bootstrapped Lempel-Ziv compression algorithm. A sequence of 10 characters originally encoded by 80 bits is compressed into 33 bits.

Bender and Wolf noticed [4] that many classes of strings, (*e.g.* written text) typically contain more than one match per window, especially if the window is long enough. They proposed a slightly different codeword structure to account for this property. Instead of encoding the length of the longest match, they find the two best matches and encode the difference in lengths between them. The longer match can be easily extracted from the shorter at a decoding stage. Finally, instead of using fixed $\lceil \log_2 w \rceil$-bit encoding for the differential match length, Bender and Wolf use a length-dependent code, in which the number of bits used to encode the number n_1 is less than or equal to the number of bits used to encode the number n_2, where $n_2 > n_1$.

The quality of applying a source coding algorithm A to document d is typically measured by its compression factor c [30], defined as

$$c(d, A) = \frac{|d|}{|d \text{ compressed by } A|} \tag{3.1}$$

where $|d|$ is the document size measured in bits.

The compression scheme used in our study is based on both the orginal paper by Ziv and Lempel and its extension by Bender and Wolf. However, we use a slightly modified algorithm, proposed by Senk [29],

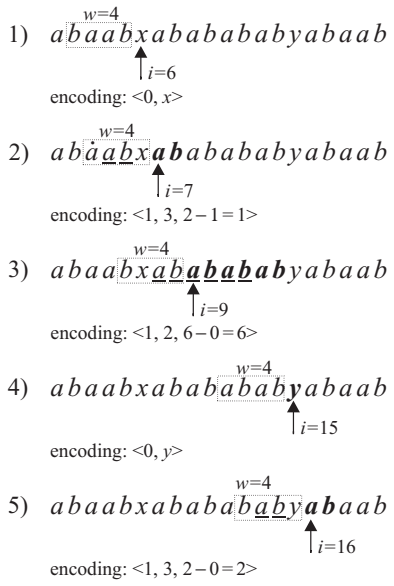

Figure 3: An example of the bootstrapped Bender-Wolf compression algorithm. A sequence of 12 characters originally encoded by 96 bits is compressed into 40 bits.

that allows for easy handling of repetitive substrings. Since the details of this encoding scheme are not widely available, we present them in Figures 2 - 3.

Consider the compression problem of an arbitrary 20-character sequence shown in Figures 2 - 3. To describe the Lempel-Ziv approach, we start at position 6, where letter x is to be encoded. Since no match can be found in the previous $w = 4$ letters, it is encoded as $\langle 0, x\rangle$, and the current position is incremented by 1. In step 2, the longest match for subsequence ab, starting at position 7, is found 3 positions to the left of subsequence ab. Thus, the length of this match is set to 2. Consequently, subsequence ab is encoded as $\langle 1, 3, 2\rangle$ and the current position is incremented by 2. In step 3, substring $abab$ from positions 9–12 is encoded using codeword $\langle 1, 2, 4\rangle$. Note that this step differs from the original Lempel-Ziv algorithm in that it allows for match extension beyond the current position i. We call this property bootstrapping [29]. However, the match length still has to be limited to w, due to the fixed encoding by $\lceil \log_2 w \rceil$ bits. In step 4, substring ab from positions 13–14 is encoded using $\langle 1, 2, 2\rangle$. Finally, in step 5, symbol y is encoded as $\langle 0, y\rangle$. The total number of bits used to encode subsequence $xababababy$ is thus 5 bits (for flags) + 16 bits (for non-encoded x and y) + 12 bits (for all encoded match positions and lengths) = 33 bits. Given that the original 10 character sequence was represented by 8 bits per character, the compression factor is calculated from Equation 3.1 as $80/33 = 2.42$.

In Figure 3, we show an example of the bootstrapped Bender-Wolf algorithm. The first difference from the Lempel-Ziv version occurs in step 2, when the current position is 7. The Bender-Wolf algorithm looks for the two longest matches within a window (matches cannot start at the same position) and then encodes the difference between them. In Figure 3 we represent the longest match using underlined characters, while the second best match is denoted by the dots above the character sequence. In step 3, the bootstrapping feature allows the match to extend to an arbitrary length, since the length difference is not encoded by a fixed number of bits. Thus, we allow a match of 6 positions, instead of 2 positions constrained by the Bender-Wolf. The total number of bits used to encode subsequence $xababababyab$ is thus 5 bits (for flags) + 16 bits (for non-encoded x and y) + 6 bits (for all encoded match positions) + 13 bits (for differential match lengths) = 40 bits. Given that the original 12 character sequence was represented by 8 bits per character, the compression factor is $96/38 = 2.52$. In this case, the same sequence would be compressed with a factor of 2.52 even using the bootstrapped Lempel-Ziv algorithm, but the sequence would be differently parsed.

Index	Code
0	1
1	0 10
2	0 11
3	00 100
⋮	⋮
L_n	code(L_n)

Table 1: code$(L_n) = \langle \{0\}^{\text{len}(\text{Bin}(L_n))-1}, \text{len}(\text{Bin}(L_n + 1))\rangle$, where $\{0\}^x$ is the size of the string of proceeding zeros, Bin(L_n) is the binary value for L_n, and len(x) is the length of string x. Bin is BCD code.

The bootstrapping feature added to the Bender-Wolf algorithm may result in having the longest and second longest match with the same length. This in turn would lead to the differential match length of zero, which is not accounted for by the original Bender-Wolf algorithm. In order to accommodate this change, we slightly modified their code, as illustrated in Table 3.1.

3.2 Datasets. A collection of 1,085 authentic papers, $\mathcal{A}$, written in English, was manually collected from several on-line journal archives accessible to our institution. In cases of some journals, only a couple of articles were freely available to us, while for the others there were no limits imposed. Our goal was to make a reasonably diverse set of authentic documents according to

Name	No. Papers	APL	APPL
Acta Biomaterialia	17	5636	3083
Elec. J. Comp. Bio.	42	7319	3948
Elec. J. Anal. Phil.	44	6489	2924
Expositiones Math.	13	3269	1375
FEBS Letters	126	8372	3560
Immunity	100	8260	4560
Injury Extra	11	2139	1141
Inter. J. Higene Health	26	5227	2799
J. Art. Intell. Res.	15	12856	5937
J. Comp. Bio.	57	9560	4333
J. Mach. Learn. Res.	48	11979	5276
Other	27	11913	5683
Nuc. Acids Res.	111	5701	3155
Ocean Engineering	5	4448	2239
Omega	8	6012	3107
Ophtamology	25	4656	2465
Optik	35	3115	1556
Ore Geological Rev.	10	8947	4754
Organic Geochem.	9	8208	4253
Organisms Div. Evo.	52	7101	3837
Organ. Dynamics	7	7702	4085
Pattern Rec. Letters	26	4624	2379
Perv. Mobile Comp.	4	9810	5338
Public Health	17	3493	1902
Religion	41	9070	4438
Topology	58	8867	3664
Urban For. Urban Green.	20	5648	3108
World Development	103	10412	5465
Zoology	29	7726	4133

Table 2: List of the journal names, number of papers from each journal, the average length of a paper from that journal before prepocessing, and the average length of a paper from that journal after preprocessing that were used in this study.

topic, style of exposition, and length. As summarized in Table 3.2, we included 28 different journal collections, and added another one, Other, consisting of randomly picked scientific papers from researchers' homepages. In the case a journal collection would allow us to retrieve a large number of papers, we stopped shortly after the 100^{th} one (to account for the entire volume).

In addition to $\mathcal{A}$, we collected 1,000 inauthentic English documents, which were obtained by querying *SCIgen - An Automatic CS Paper Generator*[3]. We refer to this collection of documents as $\mathcal{I}_{MIT}$. The papers were converted to a standard text format using Perl scripts. All downloaded papers, $\mathcal{A}$ and $\mathcal{I}_{MIT}$, were manually checked for various inconsistencies, mostly artifacts due to HTML tags, then parsed and cleaned if necessary.

3.2.1 Generating Additional Inauthentic Papers. The inauthentic paper generator described above provides only a limited subset of inauthentic papers. Thus, we generated several other classes of inauthentic papers whose purpose was to present our predictor with increasingly more difficult classification problems. The following types of documents were generated: (i) documents obtained as per-character permutations of authentic documents, (ii) documents obtained as per-word permutations of authentic documents, (iii) documents obtained by concatenating blocks of b consecutive words extracted from authentic papers, where block size $b \in \{2^i, i = 1..8\}$, (iv) repetitive documents obtained by concatenating the same block of b consecutive words extracted from authentic papers, where block size $b \in \{2^i, i = 1..8\}$, (v) documents obtained by synonym replacement from the authentic papers. These five classes of inauthentic documents are subsequently referred to as $\mathcal{I}_{char}$, $\mathcal{I}_{word}$, $\mathcal{I}^b_{block}$, $\mathcal{I}^b_{rep}$, and $\mathcal{I}_{syn}$, respectively. All classes of $\mathcal{I}$ were generated from the processed authentic documents, where the length of an inauthentic paper was equal to the length of a randomly chosen authentic paper.

Documents containing synonymous words were generated using WordNet [23], which is a downloadable lexical database that contains synonym sets of words. This technique of creating computer generated papers was considered because it attempts to preserve the content of a research paper. In general, for each word chosen to be replaced, the list of synonyms was retrieved and one was picked at random. We consider two probability distributions for synonym replacement: (i) uniform, in which any of the sybstitute words was chosen according to the same probability, and (ii) exponential, in which a more frequently used synonym was replaced with higher probability. We denote these classes of inauthentic documents by $\mathcal{I}^{uni}_{syn}$ and $\mathcal{I}^{exp}_{syn}$, respectively.

3.3 Data Preprocessing. The following steps were performed during data preprocessing. Initially, each paper was read into a string variable and all capital letters were lowercased. Words that were separated into two lines by a dash were rejoined and the dash was removed (many of these cases were performed manually during the data collection step). All non-letter characters were replaced by a white space and the string was split into an array based on space delimiting. Next, all words shorter than 2 characters and longer than 20 characters were removed, as well as the stop words, which are known to contain low amount of information. Standard Porter stemming [27] was then performed (using Lingua::Stem Perl module[4]) on the remaining words that were subsequently rejoined into a string and separated by a single space. Finally, all occurrences of multiple consecutive white spaces were turned into a single white space.

3.4 Data Represention. To construct features from the preprocessed documents we used bootstrapped versions of the Lempel-Ziv and Bender-Wolf algorithms, for various lengths of sliding windows. The rationale

[3] http://pdos.csail.mit.edu/scigen/

[4] http://search.cpan.org/∼snowhare/Lingua-Stem-0.81/lib/Lingua/Stem.pm

for such feature choice comes from the fact that the various windows, when used for document compression, have the ability to capture various long range string repetitions. Thus, combining them all together creates a *compression profile* for each document.

A set of compression factors was obtained by running bootstrapped Lempel-Ziv and Bender-Wolf algorithms and, using the notation introduced earlier, denoted as $c(d_i, blza_w)$ and $c(d_i, bbwa_w)$, where $w \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. Longer window sizes were not used in order to prevent boundary effects when training on long and short documents. This created a set of 22 numerical features that were then fed into the supervised classification algorithm. Note that we increased window sizes in powers of 2, in order to ensure efficient coding, given that $\lceil \log_2 w \rceil$ bits are necessary to encode position of the match and, for the Lempel-Ziv algorithm only, match length.

3.5 Predictor Construction and Accuracy Estimation. To build classifiers we used support vector machines, which are machine learning algorithms trained to maximize the margin of separation between positive and negative examples [31, 7]. Given that the goal of our study was not to optimize prediction accuracy *per se*, but rather to provide a proof of concept, we have only used polynomial kernels with degree $p \in \{1, 2\}$. Thus, both linear and non-linear models were evaluated. We used SVM^{light} software and its default setting for the choice of regularization parameter C [16]. Prior to SVM learning, the dataset was normalized using standard z-score normalization [13].

To evaluate predictors we used 10-fold cross-validation. We estimated sensitivity (true positive rate), specificity (true negative rate) and balanced-sample accuracy *acc* defined as

$$acc = \frac{sn + sp}{2} \tag{3.2}$$

where *sn* and *sp* are sensitivity and specificity, respectively.

4 Experiments and Results

As mentioned before, the goal of our study was to design a predictive model capable of discriminating between authentic documents and various classes of inauthentic documents.

We start with an analysis of compression performance on the various types of document classes. In Figure 4, we show compression factor for the authentic texts versus various classes of inauthentic texts as a function of th window size used in compression. A bootstrapped Bender-Wolf algorithm was used to generate the compression factor in all cases. These curves indicate that the compressibility of authentic texts lies somewhere in-between various classes of inauthentic texts. In particular, on one side of the spectrum lie compression factors of classes $\mathcal{I}_{block}$, while on the other side of the spectrum are the repetitive texts from class $\mathcal{I}_{rep}$. Typical compression factors for the $\mathcal{I}_{block}$ class ranged between 0.97 (slight expansion) and 2.1. On the other hand, the $\mathcal{I}_{rep}$ class had wide variability of the compression factors, in some cases exceeding 1000 (e.g. for $\mathcal{I}^1_{rep}$). Although we ran exhaustive experiments for both Lempel-Ziv variant and Bender-Wolf compression variant on virtually all classes of documents discussed in Section 3 Methods, we note that the compressibility behaved according to our expectations.

Next, we ran extensive experiments and estimated prediction accuracy when authentic texts were discriminated against various classes of inauthentic texts. As mentioned before, we were not particularly concerned with parameter tuning as the purpose was simply to provide evidence that the classes are separable to a large degree using the compression-based data representation. The prediction accuracies for individual features are plotted in Figure 5. Interestingly, it proved to be very easy to discriminate against the $\mathcal{I}_{MIT}$ class, even for the very small window sizes used to create features. On the other hand, the behavior of the prediction algorithms was what we expected for various other classes of inauthentic documents. For example, discriminating between classes $\mathcal{A}$ and $\mathcal{I}^1_{block}$ was easy even for the small window sizes, while discriminating between $\mathcal{A}$ and $\mathcal{I}^{16}_{block}$ and $\mathcal{A}$ and $\mathcal{I}^{256}_{block}$, the problem became increasingly more difficult for short windows, but still easily distinguishable for the longer ones. Clearly, this behavior was an artifact of the way the inauthentic classes $\mathcal{I}_{block}$ were generated, but shows that the experimental results agreed with our expectations. Arguably, the class of synonymed documents is the most difficult to recognize using compression algorithms only. This is especially true for the class $\mathcal{I}^{exp}_{syn}$, when the synonyms were taken according to the exponential distribution. In this case, the original word was often replaced by itself, which essentially created an authentic document. On the other hand, synonymed documents created according to the uniform distribution proved to be slightly easier to distinguish.

Initially, we were surprised by the prediction accuracy of 74% between $\mathcal{A}$ and $\mathcal{I}_{MIT}$ when the compression window was only $w = 2$. This accuracy increased with an increase in window size, which proved that neither short range and especially not long range pattern co-occurrences resebbled those of the authentic texts.

We also investigated the situation in which class $\mathcal{A}$

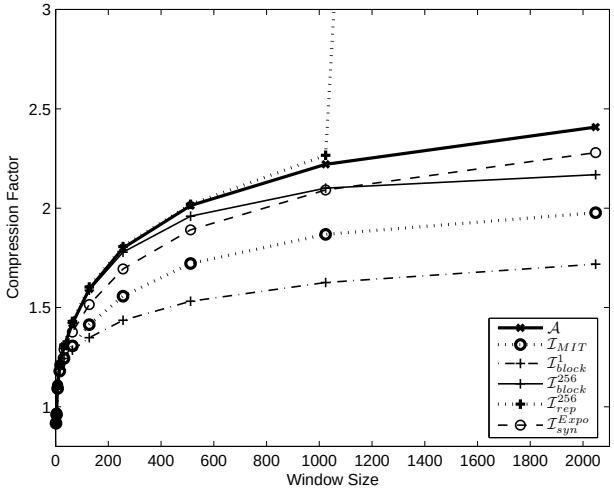

Figure 4: Compression factor as a function of the window size used for the bootstrapped Bender-Wolf algorithm. Observe that the authentic papers are bounded from below by $\mathcal{I}_{char}$, $\mathcal{I}_{block}$, and $\mathcal{I}_{syn}$ classes, while being bounded above by $\mathcal{I}_{rep}$ class.

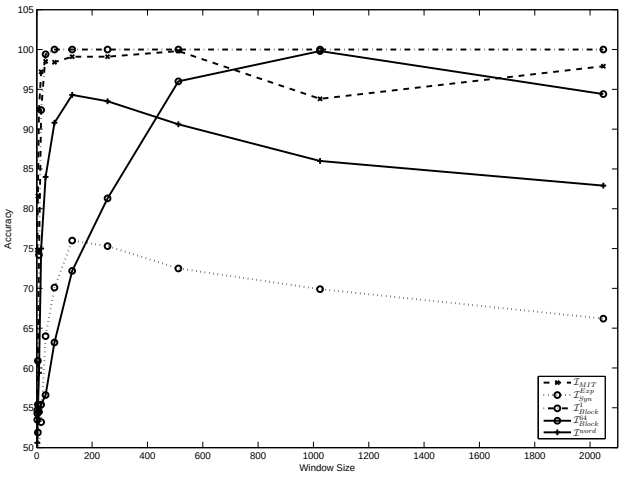

Figure 5: Prediction accuracy as a function of the window size used for the bootstrapped Bender-Wolf algorithm between authentic texts and several classes of inauthentic texts.

was distinguishable from $\mathcal{I}^1_{block}$, even for the compression window $w = 2$. Figure 6 shows the distribution of compression factors between these two classes and suggests that there was a much larger spread of compression factors for the authentic texts as compared to $\mathcal{I}^1_{block}$. For example, the minimum observed compression factor for the class $\mathcal{I}^1_{block}$ was 0.917, while the maximum was .927. Thus, the statistical properties of $\mathcal{I}^1_{block}$ were fairly stable, which does not hold true for the authentic texts. An accuracy achieved by the linear support vector machine between $\mathcal{I}_A$ and $\mathcal{I}^1_{block}$ was 54%, but increasing the degree of the polynomial to $p = 2$ lead to an accuracy of 81% for $w = 2$. Interestingly, a comparison between $\mathcal{I}_A$ and $\mathcal{I}_{MIT}$ reveals similar spreads of compression factors, but there was a shift in the average compressibility which caused these classes to be distinguishable even for the small window sizes.

Table 3 shows the classifiaction accuracy of our system for various classes of inauthentic texts and three different learning procedures. We use compression-based features combined with support vector machines of degrees $p = 1$ and $p = 2$, and a linear support vector machine constructed using the term frequency feature set. While the compression based approach was very fast and used only 22 features, the system based on the term frequency contained nearly 200,000 features. In order to train these classifiers we used feature selection filters based on the statistical tests followed by the principal component analysis. For the feature selection filters we employed standard t-test with the threshold of 0.01 (for the p-value) to eleminate the feature, while the dimensionality after the PCA was kept at 30 without further experiments.

Most of the prediction accuracies achieved by our system were close to 100%, except for the class $\mathcal{I}_{syn}$. Note that in the case of synonymed texts, the combination of all features achieved significantly better results than any of the features alone (67% vs. 82%), which justifies our approach of combining different compression algorithms in the feature set, but the increase from linear into non-linear models did not provide any improvement. Interestingly, the term frequency based approach did not have any problems with categorizing the $\mathcal{I}_{MIT}$ class either. We believe that this performance is due to the distinct terminology and a relatively narrow alphabet used for generating $\mathcal{I}_{MIT}$. However, due to the very nature of feature construction for the term frequency approach, $\mathcal{I}_{block}$ classes were essentially indistinguishable, both for SVMs of degree $p = 1$ and $p = 2$ (data for $p = 2$ not shown in Table 3). In contrast to the compression based system, the standard text categorization approach was significantly better on the class $\mathcal{I}_{syn}$. However, it is unclear whether these classifiers should or should not have achieved such high accuracies given that the synonymed classes might have preserved the content, even the flow within specific documents. It is likely that standard text categorization schemes were able to adapt on the presence or absence of specific words in $\mathcal{A}$ versus $\mathcal{I}_{syn}$.

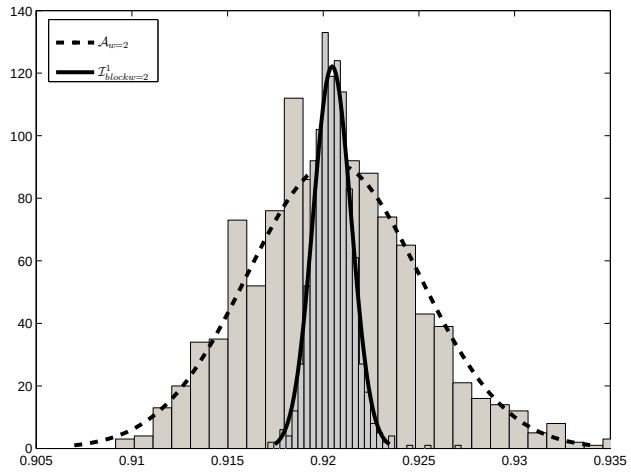

Figure 6: Distribution of compression factors for authentic papers (dashed line) and $\mathcal{I}_{MIT}$ class (solid line). The bootstrapped Bender-Wolf algorithm with $w = 2$.

Class of $\mathcal{I}$	SVM$_{p=1}$	SVM$_{p=2}$	Term Freq.
MIT	99.8%	99.6%	100%
Shuff. by char.	100%	100%	N/A
Shuff. by word	99.8%	100%	50%
$\hat{w} = 1$	100%	100%	58.6%
$\hat{w} = 16$	100%	99.9%	56.4%
$\hat{w} = 64$	99.7%	99.8%	52.2%
$\hat{w} = 256$	98.4%	96.0%	50.8%
Syn. Uniform	82.5%	81.6%	100%
Syn. Exponential	80.3%	80.3%	99.8%

Table 3: The accuracy of the predictor when distinguishing authentic papers from various forms of inauthentic papers. MIT represent the paper class $\mathcal{I}_{MIT}$. Shuff. by word represents the class $\mathcal{I}_{word}$. Shuff. by char. represntss the class $\mathcal{I}_{char}$. $\hat{w}$ represents the length of random text concatenated to form an $\mathcal{I}^{\hat{w}}_{block}$ paper. Syn. Uniform and Syn. Exponential represent the paper classes $\mathcal{I}^{uni}_{syn}$ and $\mathcal{I}^{exp}_{syn}$, respectively.

We have experimented with other classes of documents too. For example, we performed cross-validation per type of journal (29-fold cross-validation where each journal was the test set)in order to explore the influence of the style and vocabulary used in a particular journal on the performance accuracy. However, this problem also proved to be easy for the compression based system. Furthermore, we also explored the effects of length with truncated authentic papers (to length 100, 200, 400 and 800 words) versus $\mathcal{I}_{MIT}$. In these experiments, we used only a reduced set of features to avoid boundary effects, but even this situation proved to be not too difficult.

5 Discussion

In this paper we described a topic-independent supervised learning approach that can be applied to the problem of discriminating between authentic and various classes of inauthentic technical (or expository) texts. We required that inauthentic documents be machine generated in order to explore differences between human generated informative text and various possible classes of inauthentic text. Our system is based on features generated by the two related compression algorithms, bootstrapped Lempel-Ziv and bootstrapped Bender-Wolf. However, in order to account for the various long range pattern repeats, a set of cascading window sizes was used to run the software. This approach could easily be extended to a much longer list of publicly and commercially available packages in order to help to detect inauthentic documents.

It is natural to consider that due to the huge variability within the class of inauthentic papers, the detection of the authentic texts is a one-class problem. This could lead to an application of various outlier detection methods that have been extensively studied in the literature [3, 19, 6, 1]. However, for the purposes of this paper and studying whether authentic texts are separable from the various classes of inauthentic texts, we believe that it was sufficient to use supervised approaches as their performance can be quantified in well-characterized ways. Further approaches may be based on a combination of these techniques. Similarly, one could easily think of various combined models that could detect not only inauthentic documents, but at the same time be topic specific too. Such schemes could incorporate standard text categorization systems simply pre-filtered by the authentic/inauthentic preprocessor.

In general, identifying meaning in the technical document is difficult, and we do not claim herein that we have found a way to distinguish between meaning and nonsense. We do claim, however, that there are many non-trivial classes of inauthentic documents that can be easily distinguished based on compression

algorithms. The authentic documents seem to contain some hidden long-range word co-occurrences, or rather patterns, which, we argue, contribute to the flow of the paper and are a necessary condition to convey meaning in a technical document. We also argue that there might exist a class of expository texts that may not fully, or at all, convey any meaning, but still possess the flow resembling those of the authentic documents. However, it is unclear to us how these could be generated especially if they had to adhere to a topic too.

Obviously, it would be fairly easy for a human to write a meaningless article that would likely not be detected. We are not convinced, however, that on a large scale this task would be as easy for a machine, as copying valid human articles, and even substituting words by synonyms can be easily discovered using plagiarism detection techniques[25, 17].

Novel articles, however, would need to fulfill complex language models before they would qualify to circumvent even a simple algorithm like the one proposed herein. Developing an automated language system remains challenging even at the knowledge acquisition level [28], so for the foreseeable future, it may require certain effort to develop an automated generator of texts that would confuse our system, possibly augmented with serious other topic-specific modules. To support this conjecture, we developed a web interface for our system[5] and its performance can be easily tested by the scientific community.

Our initial conjecture that meaning and compression are related in informative texts raises intriguing questions about other kinds of high level structures. One can imagine varying length, format–say, as in poetry, using narrative instead of text, and so forth. How might novels be compared–one set written by humans, the other computer generated? Research has begun focusing on how to distinguish between the so-called *blog* (authentic text that usually is either a commentary or autobiographical) and non-blog. We are interested in finding out whether compressibility can say something about blogs as well. In any case, we believe that the topics scratched in this study, those related to the interplay between information, meaning and compressibility, raise interesting questions to ponder.

References

[1] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *ACM SIGMOD International Conference on Management of Data*, pages 37–46, New York, NY, 2001. ACM Press.

[2] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[3] V. Barnett and T. Lewis. *Outliers in statistical data. 3rd ed.* John Wiley & Sons, New York, NY, 1994.

[4] P. Bender and J. Wolf. New asymptotic bounds and improvements on the lempel-ziv data compression algorithm. *IEE Transactions on Information Theory*, 37(3):721–729, 1991.

[5] M. W. Berry. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer-Verlag New York, inc., New York, NY, 2004.

[6] M. Breunig, H. Kriegel, R. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

[7] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[8] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communication*, COM-32(4), 1984.

[9] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285, New York, NY, USA, 1988. ACM Press.

[10] E. Frank, C. Chui, and I. Witten. Text categorization using compression models. *Data Compression Conference*, IEEE DCC-00:200–209, 2000.

[11] Y. Gong and X. Liu. Generic text summarization using relevance measure and latent semantic analysis. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25, New York, NY, USA, 2001. ACM Press.

[12] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW2005*, pages 235–312, May 10-14 2005.

[13] J. Han and M. Kamber. *Data Mining: Comcepts and Techniques*. Academic Press, San Diego, CA, 2001.

[14] R. V. L. Hartley. Transmission of information, 1928.

[15] D. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–291, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[16] T. Joachims. *Learning to classify text using support vector machines*. Kluwer, New Jersey, 2002.

[17] M. Joy and M. Luck. Plagiarism in programming assignments. *IEEE Transactions on Education*, 42(1):126–133, 1999.

[18] D. V. Khmelev and W. J. Teahan. A repetition based

[5] www.informatics.indiana.edu/predrag/fsi.htm

measure for verification of text collections and for text categorization. In *SIGIR'03*, 2003.

[19] E. Knorr, R. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, 2000.

[20] O. Kukushkina, A. Polikarpov, and D. V. Khmelev. Using literal and grammatical statistics for authorship attribution. *Problems of Information Transmission*, 37(2):172–184, 2001.

[21] T. Landauer, D. Laham, B. Rehder, and M. Schneiner. How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. *Proceedings of the 19th annual Cognitive Science Society*, pages 412–417, 1997.

[22] M. Lesk. Automatic sense disambiguation: how to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC conference 1986*, pages 24–26, 1986.

[23] G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, Winter 1990.

[24] G. Miller and N. Chomsky. Finitary models of language users. In R. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, New York, 1963. Wiley.

[25] A. Parker and J. Hamblen. Computer algorithms for plagiarism detection. *IEEE Transactions on Education*, 32(2):94–99, 1989.

[26] R. Pasco. *Source coding algorithms for fast data compression*. PhD thesis, Stanford University, 1976.

[27] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[28] E. Rieter, S. Sripada, and R. Robertson. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intellegence Research*, 18:491–516, 2003.

[29] V. Senk. Lecture notes from the information theory and coding course., 1994.

[30] D. Solomon. *Data Compression: The Complete Reference. 2^{nd} ed.* Springer-Verlag, New York, NY, 2000.

[31] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, NY, 1998.

[32] I. Witten. Applications of lossless compression in adaptive text mining. *Conference on Information Sciences and Systems*, Princeton University, 2000.

[33] I. Witten, Z. Bray, M.Mahoui, and B. Teahan. Text mining: A new fronteir for lossless compression. *Data Compression Conference*, IEEE DCC-99:198–207, 1999.

[34] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[35] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, 1977.