A Repetition Based Measure for Verification of Text Collections and for Text Categorization

Dmitry V. Khmelev
Department of Mathematics,University of Toronto
Philological Department, Moscow State University
dkhmelev@math.toronto.edu

William J. Teahan
School of Informatics,
University of Wales, Bangor
wjt@informatics.bangor.ac.uk

ABSTRACT

We suggest a way for locating duplicates and plagiarisms in a text collection using an *R-measure*, which is the normalized sum of the lengths of all suffixes of the text repeated in other documents of the collection. The R-measure can be effectively computed using the suffix array data structure. Additionally, the computation procedure can be improved to locate the sets of duplicate or plagiarised documents. We applied the technique to several standard text collections and found that they contained a significant number of duplicate and plagiarised documents. Another reformulation of the method leads to an algorithm that can be applied to supervised multi-class categorization. We illustrate the approach using the recently available Reuters Corpus Volume 1 (RCV1). The results show that the method outperforms SVM at multi-class categorization, and interestingly, that results correlate strongly with compression-based methods.

Keywords

Text categorization, text compression, language modeling, cross-entropy

General Terms

Verification

Categories and Subject Descriptors

E.4 [**Coding and Information Theory**]: Data compaction and compression; H.2.4 [**Systems**]: Textual databases; H.2.8 [**Database Application**]: Data mining; H.3.1 [**Content Analysis and Indexing**]: Indexing methods

1. MOTIVATION

The number of texts in digital form increases rapidly and tremendously. Several profound collections are now available: for example, Project Gutenberg with more than 3500 documents; the Moshkov Library www.lib.ru for Russian-language texts (3.5 Gb as of July 2002); the Reuters Corpus Volume 1 (RCV1) (over 800K news articles); and TREC. Text collections are used intensively in scientific research for many purposes such as text categorization, text mining, natural language processing, information retrieval and so on.

Every creator of a text collection is faced at some stage with the task of *verifying* its contents. For example, it might be undesirable to have duplicate documents in the collection since they can influence the text statistics, correctness of the results obtained, consume disk space and so on. In practice, the situation with duplicates can be even more complicated, with plagiarised, expanded, corrected or template documents occurring (see Sanderson [16] and examples below in section 4.1).

We suggest a way for verifying the collection which seems to us natural, intuitively appealing and computationally effective. More precisely, we define an R-measure which is a number between 0 and 1 characterizing the "repeatedness" of the document. The R-measure is a normalized sum of lengths of all substrings of the document that are repeated in other documents of the collection. The R-measure can be computed effectively using the suffix array data structure. Additionally, the computation procedure can be improved to locate the sets of the duplicate or plagiarised documents, and to identify "non-typical" documents, such as documents in a foreign language (these documents can then be subsequently removed to ensure the collection is valid). Another reformulation leads to an algorithm that can be applied to supervised classification.

We stress that the suggested techniques are character-based and do not require a-priori knowledge about the representation of the documents, which can be, for example, UTF-encoded Unicode symbols (but we assume, of course that all documents are encoded in the same way) .

The structure of the paper is as follows. In the next section we present the definition of the R-measure. We suggest several applications in section 3, among which is supervised classification. In section 4.1, we apply the techniques to several text collections, and in section 4.2, we apply the supervised categorization method to RCV1. We make some concluding remarks in section 5.

2. R-MEASURE

Suppose that the collection consists of m documents, each document being a string $T_i = T_i[1..|T_i|]$, where $|T_i|$ is the length of the string T_i. A squared R^2-measure of document

SIGIR'03, July 28–August 1, 2003, Toronto, Canada.

T with respect to documents $T_1, \ldots, T_m$ is defined as

$$R^2(T \mid T_1, \ldots T_m) = \frac{2}{l(l+1)} \sum_{i=1}^{l} Q(T[i..l] \mid T_1, \ldots T_m), \quad (1)$$

where $l = |T|$ is the length of document T, $T[i..l]$ is the ith suffix of document T and $Q(S \mid T_1, \ldots, T_m)$ is the length of the longest prefix of S, repeated in one of documents T_1, $\ldots$, T_m. For example, let us take $T =$ "`cat␣sat␣on`" with $T_1 =$ "`the␣cat␣on␣a␣mat`" and $T_2 =$ "`the␣cat␣sat`". Then

$$R^2(T \mid T_1, T_2) = \frac{2}{10 \times (10+1)} \Big((7+6+5+4+3) + (5+4+3+2+1) \Big) \approx 0.727272 \quad (2)$$

with $R(T \mid T_1, T_2) = \sqrt{R^2(T \mid T_1, T_2)} \approx 0.852802$. Notice that in (2), the sum consists of two parts, $(7+6+5+4+3)$ from the repetition of "`cat␣sat`" $= T[1..7]$ and $(5+4+3+2+1)$ from "`at on`" $= T[6..10]$.

In principle, we could replace the sum in (1) with the maximum function to get the alternative L-measure, which is the length of the longest common substring of T repeated in T_1 or T_2 or $\ldots$ T_m:

$$L(T \mid T_1, \ldots T_m) = \frac{1}{l} \max_{i=1,\ldots,l} Q(T[i..l] \mid T_1, \ldots T_m),$$

where $l = |T|$. For the example above, we have

$$L(T \mid T_1, T_2) = \frac{1}{10} \max(7, 6, 5, 4, 3, 5, 4, 3, 2, 1) = 0.7.$$

However, we feel that R-measure is a more "intuitive" measure, reflecting perhaps that a human would assign a higher repetition rank to T than 0.7, since substrings other than "`cat␣sat`" $= T[1..7]$ are also repeated. In [10], we present several lemmas to show that the R-measure is well-behaved in many situations. In particular, we show that $R \geq L$ and the maximum value for R and L is 1.0—this will occur when a document is completely duplicated elsewhere in the collection.

$R(T \mid T_1, \ldots, T_m)$ can be computed effectively using a *suffix array*, a full-text indexing structure, introduced in [12]. It is conceptually easier to include T into the collection as $T_0 = T$ and to compute for all $j = 0, \ldots, m$ the R-measure for document T_j with respect to the rest of collection

$$R_j^2 = R^2(T_j \mid T_k, k = 0, \ldots, m, k \neq j) = \frac{\bar{R}_j^2}{|T_j|\,(|T_j|+1)/2},$$

where $\bar{R}_j^2$ is a *non-normalized* repetition measure. Then, $R(T \mid T_1, \ldots, T_m) = R_0$. The essential idea is to build a single string $S_C = T_0\$T_1\$ \ldots T_m\$$ by concatenating all documents together separated by a special sentinel symbol \$ to mark each document boundary. A suffix array is then constructed using a standard suffix array construction algorithm [4]. The $\bar{R}_j^2$ values are calculated simultaneously by reading the suffix array sequentially and adding the lengths of the longest common prefixes between adjacent suffixes for the corresponding $\bar{R}_j^2$ (see Appendix for more details). The computation of all R_j^2-values has an $O(|S_C|)$ time complexity with $O(|S_C|)+O(m)+O(M)$ memory consumption, where $M = \max_{j=0,\ldots,m} |T_j|$. This assumes that the average length of repeated substrings is significantly less than $|S_C|$ which is typical for text collections containing many documents. A heuristic approach based on so-called *resilience*, can reduce the worst case behaviour to $O(|S_C|)$ for most highly-repetitive S_C.

3. APPLICATIONS OF R-MEASURE

Statistical estimate for $\bar{R}_j^2$. Notice that in a suffix array $s_1, \ldots, s_N$, constructed for S_C (where $N = |S_C|$ and $S_C[s_j..N] \prec S_C[s_{j+1}..N]$ lexicographically for all j) all the suffixes are mixed, in the sense that the suffix array is a transposition of suffixes which is essentially random (this is reflected by suffix arrays being incompressible). If all the texts have approximately similar frequency distributions for letters, pairs of letters etc., then notice that any sequence of suffixes $s_{N_1}, \ldots, s_{N_2-1}$ where $N_2 - N_1 \approx \alpha N$, $\alpha > 0$ should contribute approximately $\alpha \bar{R}_j^2$ into the non-normalized $\bar{R}_j^2$. Hence, we can introduce a statistical estimate for $\bar{R}_j^2$:

$$\hat{\bar{R}}_j^2 = \frac{N}{N_2 - N_1} \bar{R}_j^2(N_1, N_2), \quad (3)$$

where $\bar{R}_j^2(N_1, N_2)$ is computed using $s_{N_1}, \ldots, s_{N_2-1}$. A proper discussion of the properties of the estimate (3), as well as numerical evidences for its effectiveness, is beyond the scope of this paper, but we conjecture that it can be extremely useful, especially for applying to extra large data sets and for the purpose of plagiarism detection of a single document T in a large text collection.

Locating the duplicate sets: a heuristic pruning algorithm. Duplicate documents are easily detected by determining which have R values = 1.0. Note, however, that the quantity R_j^2 above does not provide information on which document is a duplicate of another, i.e. even if we know that document T_j is completely repeated somewhere else in the text collection $T_0, \ldots, T_m$, (i.e., $R_j^2 = 1.0$), additional work is required to locate the document T_i that contains substring T_j. A very slight addition to the algorithm of computing $\bar{R}_j^2$ can help to identify the location of other repetitions. The idea is very simple: the list is organized with values $\bar{R}_{jk}^2$, containing contributions to $\bar{R}_j^2$ from suffixes of S_C starting in document k.

Clearly the memory demands could be too large to keep all $\bar{R}_{jk}^2$ in memory. We suggest a heuristic to compute approximate values for the essentially large $\bar{R}_{jk}^2$. While scanning the suffix array, a list $List(j)$ is maintained of, say, the 10 largest current contributors $\bar{R}_{jk}^2$ to $\bar{R}_j^2$. For each new contribution from suffix s_i to $\bar{R}_j^2$, we determine which document $\bar{k}$ gives the contribution and if $\bar{R}_{j\bar{k}}^2$ exists in the $List(j)$, we simply increase it by the necessary value. Otherwise we add the newly found incomplete sum $\bar{R}_{j\bar{k}}^2$ to the $List(j)$ and if the size of the list is > 10, we *exclude* the entry with the smallest sum $\bar{R}_{jk}^2$. After we have finished the scanning there is a very good chance that the $List(j)$ contains the candidates k with largest $\bar{R}_{jk}^2$ and this is born out in our experiments.

Supervised classification. Notice that there exists two distinct types of classification. By *topic categorization* (the first type of problem) one usually means assigning several possible topics to the document. The other type of problem is called *multi-class categorization*, where the document has to fall into one of several predefined classes. Of course, these types of problems are closely related, but they are definitely not equivalent. The first type of problem calls for construction of a *binary* classifier, which distinguishes only two classes and should be applied afterwards for each category. The second type of problem requires construction of a

Collection	Ref.	$R = 1.0$	$R \geq 0.5$	$R \geq 0.25$
RCV1	[15]	3.4	7.9	51.1
Reuters-21578	[14]	1.6	14.2	49.5
20news-19997	[21]	5.2	7.7	47.6
20news-18828	[21]	0.03	6.5	44.3
Russian-416	[9,11]	0.0	0.0	0.0

Table 1: % of documents that pass the specified R-measure conditions for various text collections.

multi-class classifier. R-measure can be used in this context: if one needs to select the correct class for the the document T among m classes represented by texts $S_1, \ldots, S_m$, we suggest the source be guessed using the following estimate

$$\hat{\theta}(T) = \text{argmax}_i \, R(T \mid S_i). \quad (4)$$

Identifying foreign and/or non-typical documents. Non-typical documents can be located simply by examining those documents which have the lowest R-measures, since they are the ones which are *least* repeated elsewhere in the collection and therefore candidates for rejection during a verification phase. Conversely, the articles with the highest measures will represent the ones that in same way "typify" the collection (but are also candidates for rejection because they may be duplicated or highly plagiarised). We also suggest the following method for identifying foreign language documents. In this context, we have a predominant (usually highly domain specific) language associated with the collection as a whole, and we are attempting to identify documents that have a different language to the predominant one. The method is as follows. Construct several texts, one for the language typical of the collection, S_L, and several for the target foreign languages, $S_{F_1}, S_{F_2}, \ldots, S_{F_n}$, that are anticipated to appear in the collection using text that is a representative sample for each language. For example, to identify the presence of French, German, Dutch and Belgian articles in RCV1, S_L is constructed from some sample of English text (we can use English-language documents from RCV1 for this purpose), and $S_{F_1}, \ldots, S_{F_4}$ are constructed from samples of French, German, Dutch and Belgian text respectively. The identification proceeds by using (4).

4. EXPERIMENTAL RESULTS

4.1 Analysis of various text collections

A list of R-measures was calculated for each article in various text collections using the method described in section 2. Table 1 lists the collections used and their references. **RCV1** is the extended Reuters corpus of over 800K news articles. The much smaller **Reuters-21578** containing 21578 news articles has been extensively used for benchmarking purposes in many research papers on topic categorization. 20Newsgroups has also been used extensively; it has two versions **20news-19997** and **20news-18828** based on the number of documents it contains. This collection can be used for multi-class categorization, since every message falls into only one of 20 Internet newsgroups. **Russian-416** comprises 416 texts from 102 Russian writers of the 19^{th} and 20^{th} centuries. It was prepared manually as an extension of the collection of 385 texts used for authorship attribution research using Markov Chains of letters.

Results are summarized in table 1 and plotted in figure 1 (for full lists go to `http://www.informatics.bangor.ac.uk/~wjt/Rmeasure/`). They show wide variation in the percentage of duplicates (where $R = 1.0$) from 5.2% (or 1048 articles) for **20news-19997** down to 0.03% (or just 6 articles) for **20news-18828** and 0 for **Russian-416**. Interestingly, the latter collection is the only one *not* to have evidence of significant cross-plagiarism, with the worst offender being **Reuters-21578** with 14.2 % of articles exceeding the $R \geq 0.5$ threshold. We will now highlight some further points for each collection.

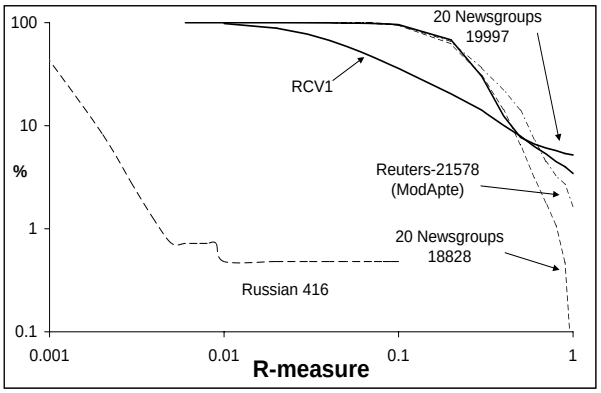

Figure 1: % of articles that exceed varying R-measures for various text collections.

Reuters-21578. This collection contains 579 duplicate documents (or 2.7%) with $R = 1.0$. However, this fact by itself does not prejudice most of the previous results obtained by researchers since benchmarking is carried out only over a subset of 10794 documents which when partitioned into a training/testing split is called the ModApte split (in fact, we present results for this subset in Table 1). Application of R-measure to the ModApte subset of the collection yields 177 (1.6%) documents repeated elsewhere. Two pairs of duplicates are shared between the training and testing splits (12495 and 18011, 14779 and 14913), and while the topics for 12495 and 18011 are the same, document 14779 omits the topic "yen" assigned to 14913. This, however, does not influence most of the published results since "yen" is not among the top 10 topics commonly experimented with. We verified that the topics for all duplicate documents in the test part are the same. Other researchers have already noted the presence of repeated documents (e.g., [4] noted that the presence of long repeated documents makes the suffix array construction difficult for this collection). We are, however, the first to highlight the existence of documents duplicated between the training and testing splits. A careful reconsideration is perhaps required of the absolute values of recall/precision and F-measures that have been reported, since duplicate documents in the test part influences the precision/recall statistics directly, while duplicate documents in the training part of the split influences the construction of the learning machine (SVM etc).

20Newsgroups. This collection has no standard split so a 10-fold or 5-fold random split with cross-validation is usually used. As noted by other researchers, **20news-19997** contains many duplicated messages (5.2%). The **20news-18828** collection was derived from this collection with the purpose of removing duplicates and empty messages. Our verification shows that it still contains six documents which are repeated in other documents. For classifiers that rely

on word-based feature extraction, these documents may become indistinguishable; for example, `comp.graphics/37261` and `comp.os.ms-windows.misc/8514` differ only by an extra new-line character. Notice that this document is assigned two different classes, which makes its correct classification impossible for multi-class categorization.

Russian-416. We found that only two documents have $R \geq 0.1$: $R = 0.168298$ for `Kabak_Soch` and $R = 0.168265$ for `Kabak_Vesn`. An inspection shows that these two novels are a journal and book version of the same text with different formatting. All other books have $R < 0.01$. Notice that this level is substantially lower than for the 20Newsgroups or Reuters collections since the average document length is much larger (284800 characters).

Reuters Corpus Version 1. A significant proportion of the articles in RCV1 are either exactly duplicated (3.4% or 27,754 articles with $R = 1.0$) or extensively plagiarised (7.9% with $R \geq 0.5$). This makes it very difficult to ensure a disjoint split of the training and testing data, which is required for meaningful evaluation. We propose the following way for overcoming this difficulty—all articles with R-measures above some threshold can simply be eliminated from *both* the training and testing data. In the case of $R = 1.0$, that leaves the remaining 97% of the collection available for use, and still leaves the possibility for researchers to perform n-fold cross-validation unlike the ModApte split which is fixed for the Reuters-21578 collection.

The large number of duplicates in RCV1 suggests another method for verifying the collection—we can examine whether the fields (such as topics, headlines and dates) have been consistently assigned to each of the duplicate articles. In an ideal world, we would expect them to be exactly or very closely matched between duplicate articles. Some variations would be expected in the case where a category judgement has been made by different human assessors possibly at different points in time (as is the case with RCV1). A poor match would be undesirable and would highlight problems that might require correction. We found that the percentage of matching fields was the following: for headlines, 56.9% matched; for dates, 78.1%; countries, 86.8%; industries, 80.1%; topics, 52.3%. The most surprising statistic is the last one—only just over half of the duplicate articles have exactly the same topics assigned to them. The percentage of matching headlines is also relatively low; this however perhaps reflects the common journalistic practice of substituting different headlines for the same story.

Although RCV1 has been designed to be a purely English-based resource, it is known that some foreign language articles are present in the collection. We applied the method described in section 3 to the problem of identifying which of the articles are non-English. Several class models were constructed from a small sampling (100-120 Kb) of English, French, German, Dutch and Belgian text obtained from a popular search engine (the methodology is described in [20]). Experiments show that good results can be achieved even with such small training texts. The method was able to find 410 French articles, 6 Dutch, 5 Belgian and one German article. Checking the articles by hand and checking against a sampling of the non-identified articles in the collection, the language identifier algorithm was found to have 100% precision with an estimated 98% recall. The list of foreign language articles that was discovered is available online at `http://www.informatics.bangor.ac.uk/~wjt/Rmeasure/`.

An examination of the documents with the lowest R-measures also highlights some further difficulties with RCV1. We found that nearly 40% of the 50 lowest scoring articles consist almost entirely of names and numbers (for example, sporting results from various world championships; and the two lowest scoring articles, 555295 and 555363, a seat-by-seat breakdown of Britain's general election). Although most of these articles are not necessarily candidates for rejection, several others do appear to be "undesirable", one article consisting almost entirely of numbers (385090) and another (385382) with a large section (2400 bytes) all in upper case. A further observation is that the articles are quite long for RCV1 (on average over 21,000 characters). This effect is related to the normalizing of R^2-measure by the squared length of the document in (1), which essentially decreases R-measure for long non-typical documents.

4.2 Multi-class categorization

We also investigated how well the R-measure performs at authorship attribution on the RCV1 data set. Authorship attribution (a multi-class categorization problem, see section 3) is an important application with many potential benefits for Information Retrieval—for example, for user modeling, determining context, a more efficient partitioning of the collection for distributed retrieval, and so on. The RCV1 collection is not primarily intended for this purpose—however, many of the articles have "bylines" assigned to them which enables us to perform authorship experiments using a much larger number of authors and larger amounts of text than is available in other authorship data sets.

Our research [19, 9, 11] has shown that using a relative-entropy based approach can be very competitive for authorship, language identification and topic categorization compared to other well-performed methods such as SVM. In fact, it was while conducting these experiments with RCV1 that we discovered the extent of the duplicate/plagiarism problems with this collection and we consequently developed the R-measure to overcome these difficulties. Perversely, we found we could also apply the measure to the same task we were investigating by using formula (4) with some interesting results as illustrated by Table 2. This table compares several approaches which proved to be effective in past experiments. Let us describe the experimental setting first and the methods used afterwards.

Experimental setting. An experimental collection was formed from 1813 articles of the top 50 authors with respect to total size of articles (notice that this set of authors differs from the top 50 authors with the largest *number* of articles). Each column in Table 2 corresponds to experiments carried out over the subsets of articles of these 50 authors, satisfying conditions for R-measure presented in the following list: condition (number of articles), $R < 0.25$ (873) $R < 0.5$ (1161) $R < 0.75$ (1255) $R < 1.0$ (1316) $R \leq 1.0$ (1813). For each subset we performed a random 10-fold split. Afterwards, 9 of the 10 parts were used for training and the remaining part was used for testing. The figures presented in the table are the percentage of correct guesses at first rank of the methods used. A description of each of the methods now follows.

R-measure. The first line in Table 2 is occupied by results obtained using formula (4).

Multi-class SVM. The well-known *panacea* in classification is SVM, which is a *binary* classifier. There are sev-

Method	$R < 0.25$	$R < 0.5$	$R < 0.75$	$R < 1.00$	$R \leq 1.0$
R-measure	82.1	86.4	87.1	87.8	89.0
Multi-SVM	80.6	83.4	83.5	84.6	85.0
Bzip2	56.9	55.2	45.9	51.9	48.2
Gzip	55.7	53.5	53.9	50.1	59.4
Markov Chains, order 1	62.3	64.6	63.2	64.3	66.1
Markov Chains, order 2	60.9	64.4	61.8	64.7	64.5
Markov Chains, order 3	48.6	60.3	59.3	61.7	63.3
RAR	**84.3**	**86.9**	**87.3**	88.5	**89.4**
PPMD, order 2	77.8	79.1	79.4	80.5	81.3
PPMD, order 3	80.6	82.3	84.0	85.0	86.4
PPMD, order 4	82.5	85.4	86.0	87.7	88.4
PPMD, order 5	82.2	86.1	86.3	**88.8**	89.2

Table 2: How well the R-measure performs at determining the top 50 authors in RCV1 compared to SVM and compression-based methods. Results are percentage of correct guesses at first rank for documents in the collection that satisfy the specified R-measure conditions.

eral approaches to reducing multi-class categorization problems to multiple binary categorization problems [1, 3, 22]. We have chosen the simplest one-vs-rest [22] SVM approach here. Feature selection was done in several steps: 1) the input text was split into "words", where the "word" is the sequence of digits or letters, the rest of the file being ignored; 2) all digits are replaced with $*$; 3) all words are down-cased; 4) the Perl implementation by its author of the Porter stemmer algorithm is applied [13]; 5) features with frequency less than 2 are dropped; 6) each feature's document frequency df is replaced with $\log(1 + df/S)$, where S is the sum of all feature frequencies remaining to this step; 7) the vector obtained is normalized with its 2-norm. The log transformation 6) is motivated by paper [5] where authorship attributions with SVM was studied from within a binary classification setting, and this was found to improve the performance by 0.5–1%. Afterwards 50 one-vs-rest SVMs were built for each author using the SVM$^{\text{light}}$ program by T.Joachims [8], and the classification was based on the maximal output of vector machines on the defined author.

Entropy-based methods. The rest of the table is occupied with several methods which use relative entropy methods for classification. The main idea is that to guess the correct author of the text T one uses the formula

$$\hat{\theta}(T) = \text{argmin}_i \, H(T \mid S_i), \tag{5}$$

where $H(T \mid S)$ is some approximation to the *relative entropy* of text T with respect to text S. Notice the similarity between (5) and (4).

The classical Shannon's definition of entropy is given in terms of Markov chains of large order [17]. This leads to the direct estimate of relative entropy using a *Markov chain* of some order k on the *letters* of the text S as a model for the source. The transition probabilities for Markov Chains are estimated from S, and $H_{MC}(T \mid S)$ is defined to be the properly normalized logarithm of the probability of the text T with respect to estimated probabilities from S [9, 11]. This estimate works well for texts of large size, but for shorter texts it is useful to combine the probabilities of symbols of text T using Markov Chains of several orders. One way of doing this is to use the PPM compression scheme [19, 20]. This provides us with the estimate $H_{PPM}(T \mid S)$, which is a properly normalized length of the compressed text T when we use a PPM model to compress S. The informatic-theoretic definition of entropy by Kolmogorov motivates the use of compression programs for estimating entropy. This naturally leads to the idea of applying *off-the-shelf* algorithms to estimate the entropy. More precisely, let us take some compressor COMP and let $C_{COMP}(S)$ be the length of the compressed text S. Then we define $H_{COMP}(T \mid S)$ as the properly normalized difference $(C_{COMP}(ST) - C_{COMP}(S))/|T|$, where ST is concatenation of texts S and T (see appendix by D.Khmelev to [11]).

Table 2 contains results of the application of estimate (5) using off-the-shelf algorithms for H_{Bzip2}, H_{Gzip}, and H_{RAR}, where `Bzip2` is a public domain program by J.Seward, using the Burrows-Wheeler transform, `Gzip` is a public domain LZ77 compressor and `RAR` is a shareware program by E.Roshal (`http://www.rarlabs.com`) which implements the PPMII compression method (a variant of PPM) by D.Shkarin [18]. We also present results for H_{MC} with order 1, 2, and 3 and results for H_{PPMD} with maximal order 2, 3, 4 and 5 using PPMD (another variant of PPM).

Let us suggest some explanation for the results in Table 2. One can see that the classification performance of good algorithms essentially decreases as we decrease the R-measure threshold. This reflects our opinion that authorship attribution is possible because authors tend to repeat themselves. The bad performance of simple Markov Chains is probably due to RCV1 articles being too short (2000–5000 characters) to yield good statistics, and the good performance of PPM shows that combining Markov Chains of different orders indeed improves results significantly. The bad performance of `Gzip` is probably explained by the short buffer it uses for compression and by the slow convergence of LZ77 to the real entropy of the sequence. The performance of Bzip2 is certainly influenced by its non-sequential block-sorting nature. The performance of RAR is comparable to those of PPMD, order 5. This is not so surprising since RAR uses a modification of the PPM.

The lower performance of SVM compared to the R-measure classifier, RAR and PPMD5, is explained (in our opinion) by SVM ignoring the relationship between words in the text. Moreover, we believe that any additional text processing would decrease performance, i.e., the text should be considered as a whole and not segmented into "words" as is done with SVM. More complicated techniques that improve the one-vs-rest approach using Error Correction Out-

put Codes (ECOC, [1]) might make SVM more competitive. However we believe we have shown that our straightforward character-based approach using R-measure, RAR or PPMD5, is impressive nonetheless.

The most striking conclusion is that the results for the R-measure classifier mirror results for RAR and PPMD5. It is well-known that PPM-type algorithms provide a very good estimate for the entropy of the text [19]. We conjecture that there exists an approximate relationship like $R \approx C/(C+H)$ between $R(T|S)$ and $H(T|S)$ for small $|T|\,/\,|S|$. The presence of such a relationship would explain the correspondence in classification results. Another corollary of the conjecture is that one gets a simple measure for similarity which is equivalent to entropy, a not so intuitive notion.

5. DISCUSSION

In this paper, we have highlighted the need for *verifying* a text collection—that is, ensuring that the collection is both *valid* and *consistent* (for example: a disjoint splitting of training and testing data is correct or easily produced; categories have been consistently assigned across duplicate documents; and undesirable foreign and non-typical documents have been eliminated).

We suggested a number of methods for collection verification using R-measure, the normalized sum of the lengths of all suffixes of the text repeated in other documents in the collection, which can be computed effectively using the suffix array data structure. The first method is based on the straightforward use of R-measure for locating duplicates and plagiarisms in the collection. The second method uses the multi-class categorization application of R-measure to locate foreign and non-typical documents with a high degree of precision/recall. The final method relies on the presence of a large number of duplicate articles—a simple check of these articles can be undertaken to determine if the fields and categories consistently match.

We applied the methods to the recently available Reuters Corpus Volume 1 (RCV1). Experiments show that over 3% of the collection is exactly duplicated elsewhere, and that many of the articles are significantly plagiarised. The implication for text categorization research is that a more careful approach is required to split the collection into training and test sets than a random or even chronological ordering. In section 4.1, we propose such an approach based on the calculated R-measures for the collection. Analysis with the collection shows that a small but significant number of articles (over 400) are non-English—these should be removed before using the collection for natural language processing. We have made available for other researchers the list of non-English and duplicate article numbers and computed R-measures—for the URLs, see section 4.1.

Further analysis of the duplicate articles in RCV1 shows that the classification made by humans is highly inconsistent. It was found that only just over half of the duplicate documents had the same topics assigned to them. This has several major implications for our studies. Firstly, such an error rate shows that the poor compression-based performance for topic categorization outlined by Frank et al. [7] is probably the corollary of erroneous classification of articles by people, violating the main assumption used by compressors: that a text comes from an ergodic source. Secondly, there exists a class of problems which is more suitable for the R-measure and PPM approach than for SVM, such as the classification of texts coming from a single source, like the papers written by a single author or in a single language. In cases where the classification depends on the presence of one or two words (as for Reuters-21578), SVM would be the preferred method. However, this might provide further evidence that the split of documents into categories for the Reuters collections is not natural as humans themselves are often puzzled by which is the correct classification of the document.

6. REFERENCES

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[2] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital SRC, Palo Alto, 1994.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

[4] A. Crauser and P. Ferragina. A theoretical and experimental study on the construction of suffix arrays in external memory. *Algorithmica*, 32(1):1–35, 2002.

[5] J. Diederich, J. Kindermann, E. Leopold, and G. Paass. Authorship attribution with support vector machines.

[6] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *41st Ann. Symp. on Found. of Comput. Sc.*, pages 390–398. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.

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

[8] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer, New Jersey, 2002.

[9] D. Khmelev. Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Text. *J. of Quantitative Linguistics*, 7(3):201–207, 2000.

[10] D. V. Khmelev and W. J. Teahan. Verification of text collections for text categorization and natural language processing. Technical Report AIIA 03.1, School of Informatics, Univ. of Wales, Bangor, 2003.

[11] O. Kukushkina, A. Polikarpov, and D. Khmelev. Using Letters and Grammatical Statistics for Authorship Attribution. *Problems of Information Transmission*, 37(2):172–184, 2001.

[12] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993.

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

[14] Reuters-21578 Text Categorization Collection. Available at `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[15] T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1—from yesterday's news to tomorrow's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31, Las Palmas de Gran Canaris, 2002. IEEE Computer Society Press.

[16] M. Sanderson. Duplicate detection in the Reuters collection. Technical report, Department of Computer Science, Univ. of Glasgow, 1997.

[17] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656,1948, 1948.

[18] D. Shkarin. Improving the efficiency of the PPM algorithm.

Problems of Information Transmission, 37(3):226–235, 2001.

[19] W. J. Teahan. Text classification and segmentation using minimum cross-entropy. In *Proc. RIAO'2000*, volume 2, pages 943–961, Paris, France, 2000.

[20] W. J. Teahan and D. J. Harper. Using compression- based language models for text categorization. In *Workshop on Lang. Modeling and Inform. Retrieval*, pages 83–88, Carnegie Mellon Univ., May 2001.

[21] The 20 Newsgroups data set. Available at `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[22] J. Weston and C. Watkins. Multi-class support vector machines, 1998.

APPENDIX

We shall now describe how to compute R-measure for all documents in collection $T_0, \ldots, T_m$ in more detail than in section 2. Suppose for simplicity that all documents $T_0, \ldots, T_m$ are non-empty. Let us introduce *sentinel symbols* $\$_0, \$_1, \ldots, \$_m$, such that $\$_i$ when sorted lexicographically preceeds any letter in $T_0, \ldots, T_m$. Also we require them to satisfy the precedence relation

$$\$_0 \prec \$_1 \prec \cdots \prec \$_m.$$

Consider the document

$$S = T_0\$_0 T_1\$_1 \ldots T_m\$_m$$

and let $N = |T_0| + \cdots + |T_m|$ and $\bar{N} = |S| = N + m + 1$. A *suffix array* for S is defined to be a sequence of integers $s_1, \ldots, s_N$, which form the set $\{1, \ldots, \bar{N}\} \setminus \{k \mid S[k] = \$_j \text{ for some } j = 0, \ldots, m\}$ and satisfy lexicographic order

$$S[s_j..\bar{N}] \prec S[s_{j+1}..\bar{N}] \text{ for } j = 1, \ldots, N-1.$$

Such an order of indexes ensures that if T_i and T_j share the common suffixes s_l and s_m, then $l < m$ for $i < j$. For a practical realization of the algorithm it is useful to define $\text{Label}(s)$ of the suffix $S[s..\bar{N}]$ to be equal to j if the prefix of $S[s..\bar{N}]$ coincides with a suffix of $T_j\$_j$. More formally

$$\text{Label}(s) = j \text{ if } S[s..\bar{N}] = T_j[k..|T_j|]\$_j T_{j+1}\$_{j+1} \ldots T_m\$_m$$

for some k. Let us now define

$$\text{Prev}(i) = \max\{k \mid k < i, \text{Label}(s_k) \neq \text{Label}(s_i)\},$$
$$\text{Next}(i) = \max\{k \mid k > i, \text{Label}(s_k) \neq \text{Label}(s_i)\},$$

which can be undefined for certain i, but for each i given either $\text{Prev}(i)$ or $\text{Next}(i)$ is well-defined. One can easily verify that the following identity for $\bar{R}_j^2$ holds; this identity can be used directly to calculate R_j^2 from the formula given in section 2:

$$\bar{R}_j^2 = \sum_{i=1}^{N} 1_{\{\text{Label}(i)=j\}} \max(\text{Lcp}(s_i, s_{\text{Prev}(i)}), \text{Lcp}(s_i, s_{\text{Next}(i)})),$$

where $\text{Lcp}(s,t)$ is the Longest Common Prefix of $S[s..\bar{N}]$ and $S[t..\bar{N}]$ if s, and t are well-defined and $\text{Lcp}(s,t) = 0$ if either of s and t are not well-defined; $1_{\{\text{Label}(i)=j\}}$ is 1 if $\text{Label}(i) = j$ and $1_{\{\text{Label}(i)=j\}}$ is 0 otherwise.

Let us now highlight several aspects of the practical realization. First, there is no need for different sentinel symbols $\$_i$, which can be replaced with a single sentinel symbol \$ (which can be the character '\000'). This can be accomplished quite easily with minor modifications to a function comparing suffixes (for example, in the C programming language, one can call `strcmp`(s,t) directly to compare suffixes $S[s..\bar{N}]$ and $S[t..\bar{N}]$, and in the case of equality, one simply needs to return the result of the comparision of s and t).

Second, a naive approach can be used to construct the suffix array $s_1, \ldots, s_N$, especially if one has sufficient internal memory (and we assume from now on that this is the case). Nowadays, RAM modules are rather inexpensive (personal computers can now be upgraded to 2 or 4 Gb) and most of the text collections can fit into the available internal memory directly (the recently available RCV1 text, for example, with XML tags removed, consumes just over 1.1 Gb). Keeping the whole collection S in the memory, we can construct the suffix array in pieces by naive Quicksort and merge them afterwards (counting sort for sorting on the first few letters can also be used for initial sort of suffixes). This algorithm exhibits $O(LN^2)$ worst case behavior and $O(LN \log N)$ average time behaviour, where L is the average Lcp of the collection. In practice, the construction of a suffix array for the 1.1Gb collection RCV1 (which we saw as containing a significant number of duplicates) requires 3-4 hours on an UltraSparc III computer using 2 Gb of memory.

Third, assuming that collection S fits in memory, in order to compute $\bar{R}_j^2$ we can simply read the suffix array $s_1, \ldots, s_N$ sequentially to compute $\bar{R}_j^2$. The only catch here is that in order to deal with Prev and Next indexes with different labels we should maintain a stack of subsequent suffixes with coinciding labels. If the next suffix has a label different from the one on the top of the stack (j, say), the stack can be emptied contributing the Lcp's to $\bar{R}_j^2$. This process is extremely effective since one takes advantage of long sequential reads from the hard disk, which are much more effective than random access reads. In the worst case, the stack requires $M = \max(|T_0|, \ldots, |T_m|)$ memory cells, but on average, only a few of them are used. The time complexity for computing $\bar{R}_j^2$ is usually significantly less than the time required for naive suffix array construction as described above. For example, processing of RCV1 requires less than hour on UltraSparc III computer using 1.5 Gb of memory. If computation takes too long, one is better to stop it and detect the long duplicated documents using estimate (3).

Finally, if the text collection does not fit into RAM, one can split it into several parts, say, A, B, C and apply the technique described to pairs AB, BC and AC. This still allows one to identify duplicated and essentially plagiarized articles. Of course, there is a trade-off between memory and time complexity, but given the memory resources currently available, it is possible to verify in reasonable time a great percentage of the existing text collections.

As one can see, the method for computing R-measure is straightforward and easy to program. We provide programs for computing R-measure at the same URLs given in section 4.1. The interested reader is redirected to [10] for further details.

At the end we wish to discuss the method for storage of suffix array. It is well-known [2] that text can be stored using its Burrows-Wheeler transform (BWT), which can be further compressed, up to 0.25 of the size of the source (natural language) text. Interestingly, the suffix array of the text can also be easily restored from BWT. Together with recent results on the possibility of search inside compressed BWT text [6], our results on R-measure classification performance open an interesting way for keeping text collections using a compressed BWT representation with preserving the possibility for search and classification.