Evolving Better Stoplists for Document Clustering and Web Intelligence

Mark P. Sinka, David W. Corne[1]
Department of Computer Science, University of Reading, Reading, RG6 6AY, UK
m.p.sinka@reading.ac.uk, d.w.corne@exeter.ac.uk

Abstract: Text classification, document clustering and similar document analysis tasks are currently the subject of significant global research, since such areas underpin web intelligence, web mining, search engine design, and so forth. A fundamental tool in such document analysis tasks is a list of so-called 'stop' words, called a 'stoplist'. A stoplist is a specific collection of so-called 'noise' words, which tend to appear frequently in documents, but are believed to carry no usable information which would aid learning tasks, and so the idea is that the words in the stoplist are removed from the documents concerned before processing begins. It is well-known that the results of document classification experiments (for example) are invariably considerably improved when a stoplist is employed. Current stoplists in regular use are, however, rather outdated. We have explored this claim in recent work which produced new stoplists based on word-entropy over modern collections of documents. In this work we introduce the notion of *optimising* a stoplist, and use stochastic search in conjunction with *k*-means clustering to converge on stoplists which lead to better performance on certain tasks than any previously published.

1. Introduction

Text classification, document clustering and similar document analysis tasks are currently the subject of significant global research (Aas & Eikvil [1], Sebastiani [2], [3]) since such areas underpin the enterprises of web intelligence, web mining, web search engine design, and so forth. A fundamental aspect of almost all machine-learning tasks involving document processing is a list of so-called 'stop' words, called a 'stoplist'. A stoplist is a specific collection of so-called 'noise' words, which appear frequently in documents, but are believed to carry no usable information to aid learning tasks. The idea is that the words in the stoplist are removed from the documents concerned before processing begins. For example, in the standard vector-space representation of a document, it might be thought useful to include the word 'chess' as a feature in a vector of specific word frequencies when the task concerns clustering of search engine results into themes (such as games and pastimes). However, the word 'and' is unlikely to be helpful as an element of the feature vector in this or almost any task; and more to the point, including such information-poor words will typically lead to degradation in accuracies.

Stoplists aren't a new idea; in fact it can be argued that current stoplists are rather outdated. One of the most commonly used stoplists, for example, is that of Van Rijsbergen [5], which is now almost 30 years old. Fluctuations in word use over time, as well as the emergence of web-specific words (either as a bias in the arena for document analysis tasks, or in general usage anyway) both suggest a hypothesis that document analysis tasks could be better served by more up to date stoplists. We have explored this claim in recent work,

[1] Now at the Department of Computer Science, University of Exeter, UK.

producing new stoplists based on word-entropy over modern collections of documents [6]. In this work we introduce the notion of *optimising* a stoplist, and use stochastic search in conjunction with *k*-means clustering to converge on stoplists which lead to better performance on certain tasks than any previously published.

The main real-world issue to which this work relates is web classification. Ideally, web search and general web-exploitation will utilise categorical information about web document; however, such categorisation can only really be done by automated methods, which will probably rely on unsupervised learning methods which in turn will use stoplists. Non-automated methods of web categorisation are arguably quite infeasible. For example, by taking the search engine with the highest index, currently Google [7], a suitable lower bound for the size of the web can be ascertained. Google [7] currently indexes over 3 billion web documents. With an index of this size, manual back classification of the web would be hugely resource intensive, even for the categorization of each of the estimated 1.5 million web documents added each day [8]. Various suggestions for document classification by the document's author have been put forward, but with no suitable taxonomy of the web, along with previous experience of META keywords/description tag misuse, this would be an ineffective method. For these reasons, most researchers in this field agree that total classification of the web is only achievable through the use of total or partially automated categorization methods.

We focus on unsupervised clustering, because it doesn't require *a priori* information. This is extremely important in the context of web document categorization due to the fact that any humanly generated taxonomy consisting of the many categories of the web would be highly subjective, never mind hard to produce, therefore bringing into question the usefulness of supervised categorization techniques. As we have mentioned, stoplists are a fundamental tool for improving the accuracy of categorization techniques in information retrieval (and associated branches such as unsupervised clustering). The use of stoplists is justified by research suggesting that over 50% of all words in a small typical English passage are contained within a list of about 135 common words [9]. These can be considered to be noise words as suggested by Van Rijsbergen [5], and should be removed as part of any pre-processing in text analysis experiments.

Stop words have been fairly well researched and a handful of lists have been published, but these lists were compiled on the basis of particular experiments on text retrieval. Many researchers, ourselves included have used these as standard stoplists, finding indeed that they greatly aid in achieving effective clustering or classification, however their age and the context of their compilation must bring into question their continued general usage.

In particular, a large part of modern research in document clustering is concerned specifically with web documents, and it can be argued that a web-specific stoplist would be beneficial. During our experimentation in [10], we noted that the text retrieval stop-word lists were lacking some very common web-specific terms such as 'email', 'contact', etc; this is of course related to the fact that these lists were developed before the mass expansion of the web in the early 1990s.

Our recent work [6], attempted to remedy these shortcomings of existing web-naive stoplists, and we proved using information entropy that these long standing stoplists can be improved when applied to the web document categorisation domain. Encouraged by the results, and noticing the variety, diversity and unpredictability of the relationship between stoplist size, pedigree and results – we were led to cast the task of finding a good stoplist as an optimisation problem, and hence to consider applying black-box optimisation techniques such as evolutionary computation (Fogel et al, [11]; Holland, [12]; Koza, [13]). This paper records our first steps towards optimising stoplists in this fashion.

In section 2 we outline existing 'industry standard' stoplists and briefly point to ongoing work aimed at improving them via information theoretic notions. We also outline

the selection process that determined the pool of candidate words available for stoplist optimisation. Section 3 covers our optimisation algorithms, in particular describing the fitness function which attempts to evaluate the quality of a stoplist. Our experiments and corresponding results are described in sections 4 and 5 respectively. Finally we summarise and discuss in section 6.

2. Current Stop Lists

Probably the most widely used stoplist in circulation today is the list of commonly occurring words published by Van Rijsbergen in his seminal book on information retrieval [5], and this is most commonly referred to as the *Van Rijsbergen stoplist.* Despite the fact that this book was first published in 1975, the original list still serves well even when applied to more 'modern' tasks such as unsupervised clustering of web documents [10]. The other commonly used stoplist is the *Brown stoplist*, so called because it originated from the Brown's document corpus [4].

We showed in recent work [6], that these existing stoplists could be improved upon. In that work, word entropies were calculated over each of two datasets (the BankSearch Dataset [10], and a new 'Random' dataset [6]), and these were used to generate two ranked lists of words in ascending order of entropy, leading to various opportunities to construct new stoplists. Even though the Van Rijsbergen and Brown stoplists generally achieved the highest and most robust accuracies over a range of tasks, new stoplists occasionally achieved better accuracies. These new stoplists were either based on the entropy lists (e.g. the top 300 words from one of the lists) or unions of the standard stoplists with a top slice of one of the entropy lists.

In the present work our aim was to evolve optimal stoplists, and this requires a pool of words to choose from as potential candidates. We chose to piggyback on the work done in [6] for this task, and used a pool of 870 distinct words which arose from the union of the Van Rijsbergen stoplist, the Brown stoplist, the top 500 entropy-sorted words from the BankSearch Dataset, and the top 500 entropy-sorted words from the Random Dataset (introduced in [6]). The decision to select the top 500 from each of the entropy-sorted lists was a trade-off between excluding potential words (if a smaller selection was chosen), compared with computational time needed to perform enough iterations to generate useful stoplists if too many words were selected. Therefore we felt that 870 words was a reasonable number of candidate words that balanced the trade-off well.

3. Evaluating Candidate Stoplists: The **Fitness Function**

The most costly element of this research, both in terms of development and experimental time, was the fitness function needed to evaluate each prospective stoplist. The evaluation time for each candidate stoplist is so great due to the numerous mandatory steps that must be carried out to generate a corresponding accuracy measure. The basic idea of the fitness function is to estimate the quality of a stoplist by performing a collection of non-trivial machine learning tasks supported by that stoplist. This is necessarily time-intensive for several reasons. The machine learning task needs to be non-trivial, otherwise we would have no confidence in the general usefulness of stoplists which performed well. Multiple trials of the same task are needed within a *single* evaluation, since we are using a non-deterministic clustering method (as is quite common). However, the main reason for the time complexity is the data pre-processing activities following the interpretation of a specific stoplist. What is normally done at a pre-processing stage (and then usually never again for that set of

documents), has to be done in *every* fitness evaluation here, and that is the removal of the words specified in the stoplist from the documents that will form the basis of the *k*-means experiments. We describe this in a little more detail next.

The fitness evaluation function can be split into two main sections; the feature extraction and document representation; and the unsupervised *k*-means clustering. The first step is to parse each web document in turn to select the set of words that appear at least once in each document. It is at this stage that the candidate stoplist is used to remove all words that are present in the stoplist from the selected set of words. This process was repeated for each of the 2000 documents involved in the clustering task (see section 4). An additional procedure that is often included at this point as part of web document clustering is the combining of words with similar word stems (word stemming). This allows for very similar words to be classed as being the same, just as a human would see the words "germany", "german", and "germans" as being from the same root word, which in this case is "german". Stressing simplicity first, we opted *not* to employ word stemming for our initial research both due to fact that the extra time would increase the already high iteration cycle time significantly. Also, word stemming has been shown to be not universally helpful [10].

The second step was to combine the union of each of the 2000 sets of extracted words into a global set that comprised all words occurring over all documents (obviously minus the stop words), this set would become the basis for the specific vector representation used in this individual fitness evaluation.

The next step was to create a master word list from the set of combined words (described above), containing every word in the combined set along with its associated frequency; this list was then sorted according to descending word frequency. Then we cut down the master list to contain only the top 2% of the most frequently occurring words. The pruning of the master list is performed to balance the trade-off between document recall and unnecessary noise. By using the entire master vector we would be effectively representing documents using insignificant words that may only appear in a few documents, but by using too small a percentage of the master list we would only be including words that occur very frequently in all documents, these words carry very little entropy and hence are not representative of individual documents. Obviously using the stoplist reduces the amount of poorly representative words, as without stoplist removal, noise words would occupy the top 300-500 entries in the sorted master list. Finally, a feature vector v_i was created for each document i, such that the jth element in v_i was w_{ji}/s_i, where w_{ji} is the number of occurrences in document i of the jth most frequent word in the combined set of extracted words for all 2000 documents, and s_i is the total number of words in document i.

The second part of the fitness evaluation involved performing *k*-means clustering on the set of 2000 document features. The fitness value is generated by clustering the 2000 documents into 2 clusters, because we are using a pre-classified dataset (see section 4), it allows us to ascertain the accuracy of each cluster run compared to each document's assigned category. We set k=2 for the clustering experiments so that it matched with the number of categories contained in the 2000 documents. We also performed 50 cluster trials for each candidate stoplist, selecting the highest achieved accuracy over the 50 trials to be the resulting fitness value.

Our implementation of *k*-means clustering was standard, although certain issues tend to vary between implementations and we clarify those here. *K*-means begins by generating random vectors to act as initial cluster centres. Each required random cluster centre was created by copying the contents of a randomly chosen document feature vector from the vector space. Another important issue is the treatment of 'dead' cluster centres (containing no documents, since all vectors are closer to some other cluster centre). We chose to do nothing when a cluster centre died, in case a vector became re-assigned to it in the future.

Again, this stressed simplicity, although other options are generally more favourable, especially when *k* is low.

4. Experiments

The issue of selecting suitable words for candidate stoplists can be classed as an optimisation problem. We began our exploration of optimising stoplists by using evolutionary algorithms ([11], [12] and [13]). Our initial choices of algorithm are motivated by the need for algorithms which can optimise quickly – i.e. in relatively few iterations, and so we use hillclimbing (the simplest possible evolutionary algorithm, with a population of 1), and a straightforward steady-state evolutionary algorithm with binary tournament selection, mutation only (no crossover), and a population of 10. We will refer to these algorithms hereafter as HC and EA respectively.

In order to compare HC and EA, each one was set the same task, which involved extracting and clustering two content-distinct categories from the BankSearch dataset ([10]; www.pedal.reading.ac.uk/banksearchdataset). The two categories chosen were Commercial Banks (A) and Soccer (I).

Both HC and EA use a simple chromosome to represent candidate stoplists, where the chromosome is simply a binary string. A candidate stoplist is generated from the given chromosome simply by comparing each chromosome element with the corresponding element in the master list of candidate stop words. The generation of this master set of stop words was described in section 2.

If the element in the chromosome is a '1' then the corresponding word from the master list of candidate stop words was added to the list of words forming a candidate stoplist, if it was a '0' then the word was not added.

The hill-climbing algorithm, which is probably the simplest optimisation algorithm to perform, simply generates a candidate stoplist via repeated mutation of the chromosome, evaluating mutants using the fitness function outlined in section 3, and keeping the mutant as the new 'current' solution if it is better or equal to the incumbent (otherwise discarding it). Again stressing simplicity, we employed single point mutation, where one element of the chromosome was chosen at random and flipped, corresponding to either removing or including a new candidate stopword.

Due to the time complexity of the fitness function, 1000 iterations was the effective time limit given to the algorithm for it to generate a useful stoplist. In order to get the maximum performance from the relatively small number of iterations we decided to seed the initial chromosome so that the algorithm started out with an encoding of the Van Rijsbergen stoplist.

The EA used a population of 10 candidate chromosomes. In each iteration, a chromosome was selected with binary tournament selection, mutated using the single point mutation used for HC, and then evaluated using the same fitness function. If the resulting accuracy of the newly generated candidate chromosome was greater than or equal to that of the worst chromosome currently in the population, then the new chromosome was added to the population and the worst (or, a random worst) was removed. This was also repeated for 1000 iterations, corresponding to 1,000 fitness evaluations.

5. Results

To place the results in context, we show part of the most closely related set of experiments from [6] in Table 1. This shows the results of 2-means experiments on categories A and I

from the BankSearch dataset. The standard stoplists appear in the first two rows, and a selection of other stoplists derived from that work are included. Note the fact, commonly found in [6], that the standard used datasets are strong-performing and robust, while the use of a stoplist is generally significantly better than not using a stoplist at all. It is also notable that there seems to be an optimal size for a stoplist (the Van Rijsbergen and Brown stoplists contain 319 and 425 words respectively). Simply including large numbers of words with low entropy does not improve the results.

Table 1: Clustering two "distinct" subject categories from the BankSearch Dataset: "Commercial Banks" and "Soccer", with a fixed feature vector length of 2% - with SLR referring to a stoplist derived from entropies of words from a specially devised Random Dataset (taken from [6]).

Stoplist	Mean	Median	Best
Van Rijsbergen	73.7	91.3	94.4
Brown Stoplist	72.5	91.3	94.4
No Stop List	61.1	64.5	66.4
Top 10 of SLR	73.1	87.6	89.1
Top 50 of SLR	66.8	51.2	**94.8**
Top 100 of SLR	69.5	52.5	93.9
Top 150 of SLR	72.0	89.2	93.6
Top 300 of SLR	68.4	55.9	91.8
Top 500 of SLR	68.9	57.8	90.2
Top 1000 of SLR	67.5	57.5	89.6

The starting point for the experiments in this paper was always the Van Rijsbergen stoplist itself. However, for speed reasons we used a different document representation in which the feature vector contained just the top 1% in the ranked list of most frequently occurring words in the combined categories A and I. The median performance of the Van Rijsbergen stoplist turned out to be 93.05% (rather than 91.3% above, which used a feature vector of length 2%).

We first report on the results of two sets of ten trial runs each, one set for hillclimbing (HC) and one set for the simple evolutionary algorithm (EA). In the hillclimbing runs we always started off with the Van Rijsbergen stoplist, and the population of 10 in an EA run initially contained one perfect copy of the Van Rijsbergen list, and 9 mutants of it, where the mutants were obtained by randomly changing 10% of the representing bitstring. We stress that the considerable computational intensity of the enterprise of optimising stoplists is in no sense a barrier to usefulness, since this is a *design* problem. That is, stoplist design itself does not need to be performed quickly, but the result – a well-designed stoplist – could be of lasting value in improving the accuracy and sensitivity of document analysis tasks in many spheres. In an additional experiment, eight longer EA runs were performed (5,000 iterations each). Table 2 summarises all of the results, which are described in the following.

5.1 The Hillclimbing Experiments

The best and worst trials in the hillclimbing experiment produced stoplists which led to accuracies on the classification task of 95.35% and 96.1%. These improve upon the Van Rijsbergen stoplist by 2.35% and 3.1% respectively. Even the worst hillclimbing result was better than any stoplist used or derived in Sinka & Corne (2003) which found many stoplists to compete with Van Rijsbergen's and Brown's stoplist by using entropy based measures. For convenience (and also available via http://www.pedal.rdg.ac.uk/banksearchdataset/), we

show one of the optimised stoplists in Figure 1. The ordering is an artefact of our methods, which order words based on each of frequency, lexicography and entropy. The words in **bold**face are words which were added during the hillclimbing run (since they either led to increased accuracy, or their inclusion did not degrade accuracy); each of which is not part of the Van Rijsbergen stoplist. All other words are in both the optimised and Van Rijsbergen stoplists. An interesting collection of words is those in Figure 2. Each of this is a word whose removal during the hillclimbing process led to an improvement (or, no degradation) in accuracy; each of these is in the Van Rijsbergen stoplist, but hence not in our optimised stoplist.

a,about,across,after,again,against,alone,already,also,although,among,an,and,another,**anybody**,anywhere,are,**area**,**areas**,as,at,**backing**,**backs**,become,becomes,behind,being,both,by,can,cannot,**cases**,**come**,could,**d**,**differ**,**different**,do,done,down,**downed**,**downs**,each,either,**ending**,**ends**,enough,even,**evenly**,ever,every,everything,everywhere,**far**,**finds**,first,for,four,from,full,further,**furthered**,**furthers**,**gave**,**generally**,**gets**,**good**,**goods**,**great**,**greater**,**grouping**,**groups**,**h**,had,have,he,herself,here,**higher**,him,himself,his,however,,interest,**interests**,into,is,it,**j**,**just**,**k**,keep,**kind**,**known**,**l**,**largely**,**later**,**latest**,least,less,**like**,made,**making**,many,may,**member**,**members**,might,more,mostly,much,must,my,myself,**necessary**,**need**,**needing**,never,**new**,**newest**,not,nobody,noone,**numbers**,of,off,often,**older**,**oldest**,on,once,one,only,or,our,out,over,part,**parting**,perhaps,**places**,**pointing**,**presented**,**puts**,**right**,**said**,same,**saw**,**say**,**second**,see,seem,seemed,several,she,**showing**,side,since,**smallest**,so,someone,states,still,**sure**,take,**taken**,than,the,their,them,therefore,these,they,**thing**,**thinks**,those,**thoughts**,three,through,thus,to,**took**,toward,**turn**,**turning**,**uses**,**used**,**wanted**,**wants**,was,we,**went**,were,what,when,where,while,whole,will,with,within,without**workedworkingworks**would**,y,**you**younger**your**,z,10,information,page,free,following,20,available,set,25,11,16,using,current,please,below,18,day,web,based,name,click,rights,email,people,return,list,line,24,note,provide,100,main,17,times,complete,29,previous,development,online,re,link,check,21,post,includes,limited,40,days,currently,found,easy,related,similar,building,services,range,view,five,run,specific,33,add,level,team,move,example,called,uk,looking,value,1999,pages,takes,26,course,am,ve,offer,performance,plus,live,particular,visit,designed,via,class,directly,quality,six,etc,55,receive,usually,continue,basic,37,90,wide,website,book,reason,links,offers,amount,52,soon,providing,added,42,2003,national,events,updated,version,friend,story,technology,articles,city,john,university,process,recent,buy,issues,government,reports,house,food,gt,phone,00,america,advertising,text,water,care,subscribe,music,feel,feedback,according,1996,industry,tools,result,afterwards,amongst,becoming,beforehand,beside,beyond,cant,con,de,describe,eg,elsewhere,fifteen,fire,former,formerly,forty,front,hasnt,hereafter,herein,hereupon,hers,hundred,indeed,latter,ltd,mill,mine,moreover,namely,neither,nevertheless,nine,onto,ours,sixty,somehow,sometime,sometimes,themselves,thence,thereafter,therein,thin,throughout,twelve,un,whereafter,whereas,whereby,whither,whom,yourself

Figure 1: The best optimised stoplist so far, produced after a 1,000-iteration run of Hill-climbing, measured via median accuracy of 20 2-means trials for separating the "Soccer" and "Commercial Banks" categories in the BankSearch Dataset.

above, all, almost, along, always, any, anyone, anything, around, back, be, became, because, been, before, between, but, during, everyone, few, find, get, give, go, has, her, how, in, its, itself, last, me, most, never, next, no, nothing, now, nowhere, other, others, per, put, rather, seeming, seems, should, show, some, something, somewhere, such, that, then, there, together, too, two, under, upon, us, very, well, whether, which, who, whose, why, yet, yours

Figure 2: Words in the Van Rijsbergen stoplist which do not appear in the optimised stoplist.

Consider the word 'in' which seems like a classic candidate for a word in a stoplist, and is indeed in the Van Rijsbergen list, however it does not appear in our optimised stoplist. In fact, its removal (as a mutation within the hill-climbing run) caused in context an improvement in accuracy from 94.85% to 94.90%. Other aspects of Figure 2 seem to reflect current usage. While 'anyone' and 'everyone' are in the Van Rijsbergen stoplist but not the optimised stoplist, the opposite is true of 'anybody'.

The many words which were included during the hill-climbing process include several which seem to be world-wide-web orientated. This is a natural result of our bias towards datasets of web documents, and continues our argument (started in [6]) that web-intelligence applications should employ web-specific stoplists. Examples of these words are *web*, *website*, *email*, *online* and *link*. Email, for example, is a frequently occurring word in

web documents, but intuitively carries no usefully discriminative information about a document's content.

5.2 The Evolutionary Algorithm Experiments

The best and worst trials in the 1,000 iteration EA experiments were 95.2% and 96.15% respectively. The EA clearly has a slightly larger range in performance than the HC method, and a slightly lower mean performance (95.58% versus 95.66%), and a slightly lower variance, but the achieved the best result from all of the 1,000 iteration experiments (96.15% accuracy compared with the 93% achieved by the Van Rijsbergen stoplist). A T-test reveals no statistically significant difference between the two approaches. The longer EA runs achieve the best mean performance, and the best overall result (96.2%), however, again, there are insufficient data to lead to statistically significant conclusions from a T test. The best result overall achieves a 3.2 points improvement in accuracy over the van Rijsbergen stoplist, and all experiments managed to improve upon the Van Rijsbergen list by at least 1.8%.

Table 2: Results of the hillclimbing (HC, 1000 iterations), evolutionary algorithm (EA, 1000 iterations) and evolutionary algorithm (EA, 5000 iterations) experiments. The final row shows mean values.

HC	EA	EA (5000)
95.8	95.55	95.95
96.1	95.75	95.5
95.35	95.2	96
95.65	95.4	94.8
95.8	**96.15**	96.05
95.65	95.4	95.95
95.7	95.9	96.05
95.35	95.6	**96.2**
95.6	95.45	-
95.6	95.35	-
95.66	95.58	95.81

6. Conclusions and Future Work

The use of stoplists is fundamental to document analysis and learning tasks, and is hence particularly important in underpinning web-intelligence activities. However, the classic stoplists traditionally in use are arguably outdated and not sufficiently web-specific. In previous work we have derived new stoplists based on information theoretic measures calculated on the BankSearch dataset and a random dataset [6]. The stoplists produced and tested in that work included the top *n* lowest calculated entropy words for various *n*, as well as Van Rijsbergen's, the Brown Stoplist, and various combinations of those mentioned. Here, in contrast, we have started with a union of the entropy-based stoplists and Van Rijsbergen's stoplist, and improved upon that via simple optimisation methods. The computational cost of optimisation in this context (each evaluation of a stoplist taking a number of minutes to perform) means that as yet we have little to go on as regards statistical analyses of the methods employed, however we certainly have results in the form of stoplists which perform better than any from [6] on a certain task.

In what we can report on so far, we hence believe we have proven the feasibility of producing better stoplists via optimisation. Stoplists have been produced which, when tested

on a particular classification task, perform better than the standard stoplists do on that task, and better than several other stoplists derived via information theoretic means in [6].
The main issue we have yet to examine is the degree to which the optimisers are specialising on the particular choice of categories used. Consider separating two similar categories of documents, such as "Banks" and "Insurance Companies"; in this case, "money" may be a valid noise word which would be beneficial to have in a stoplist (since, if not, and hence included in the feature vector, this adds noise to the classification task since it is not a usefully discriminative feature for the task at hand). Our choice of two distinct categories in the present work was done to minimise the chance of including such words in the stoplist. So, we expect that a stoplist optimised to perform well in separating two distinct categories would generally perform well in other contexts. But, with many subtle effects at play here, our initial tests of the optimised stoplists in other contexts have not outperformed the Van Rijsbergen stoplist in those contexts). Specialised stoplists have their uses (for example, in clustering documents towards their placement in an existing categorisation or hierarchy), and the work reported herein suggests that it is possible to optimise stoplists for such purposes, however to achieve more generally usable optimised stoplists requires further research, which is likely to use a more sophisticated fitness function which tests the stoplists performance in a wider variety of tasks.

Acknowledgements

The authors thank SEEDA (the South East England Development Agency), the EPSRC (Engineering and Physical Sciences Research Council), and Simon Anderson and others at BankSearch Information Consultancy Ltd, for ongoing financial support for this research.

References

[1] Aas, K., Eikvil, A. (1999) *Text Categorisation: A survey*, Technical report, Norwegian Computing Center, June, <http://citeseer.nj.nec.com/aas99text.html>

[2] Sebastiani, F. (1999a) *Machine learning in automated text categorisation: A survey*. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, C.N.R., Pisa, IT, 1999. . <http://citeseer.nj.nec.com/article/sebastiani99machine.html>

[3] Sebastiani, F. (1999b) A Tutorial on Automated Text Categorisation. In Analia Amandi and Ricardo Zunino, editors, Proceedings of ASAI-99, *1st Argentinean Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, AR, 1999. <http://citeseer.nj.nec.com/sebastiani99tutorial.html>

[4] Fox, C. (1992) "Lexical Analysis and stoplists", in *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, 1992. (including the 'Brown Corpus' stoplist).

[5] Van Rijsbergen, C.J. (1975), *Information Retrieval*, Butterworths.

[6] Sinka, M.P., Corne, D,W. (2003) "Towards Modernised and Web-Specific Stoplists for web document analysis", *Proceedings of IEEE Web Intelligence 2003*, Halifax, Canada, (To appear).

[7] Google Search Engine, <http://www.google.com>

[8] Pierre, J.M. (2000), "Practical Issues for Automated Categorization of Web Sites", September 2000. <http://citeseer.nj.nec.com/pierre00practical.html>

[9] Hart, G.W. (1994), "To Decode Short Cryptograms", *Communications of the ACM*, 37, 9, September 1994, 102-108.

[10] Sinka, M.P., Corne, D.W. (2002) A large benchmark dataset for web document clustering, in Abraham, A., Ruiz-del-Solar, J., Köppen, M. (eds.), *Soft Computing Systems: Design, Management and Applications, Volume 87 of Frontiers in Artificial Intelligence and Applications*, 2002, pp. 881-890 (ISBN: 1 58603 297 6).

[11] Fogel, L.J., Owens, A.J., and Walsh, M.J. (1966). *Artificial Intelligence Through Simulated Evolution*, John Wiley, New York.

[12] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[13] Koza, J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. *MIT Press*, Cambridge, MA.