2008 Eighth IEEE International Conference on Data Mining

Multi-label Classification using Ensembles of Pruned Sets

Jesse Read, Bernhard Pfahringer, Geoff Holmes
Department of Computer Science
University of Waikato
Hamilton, New Zealand
jmr30,bernhard,geoff@cs.waikato.ac.nz

Abstract

This paper presents a Pruned Sets method (PS) for multi-label classification. It is centred on the concept of treating sets of labels as single labels. This allows the classification process to inherently take into account correlations between labels. By pruning these sets, PS focuses only on the most important correlations, which reduces complexity and improves accuracy. By combining pruned sets in an ensemble scheme (EPS), new label sets can be formed to adapt to irregular or complex data. The results from experimental evaluation on a variety of multi-label datasets show that [E]PS can achieve better performance and train much faster than other multi-label methods.

1 Introduction

The traditional data mining task of *single-label* classification, also known as *multi-class* classification, assigns each instance d a single label l from a previously known finite set of labels L. A single-label dataset D is composed of n instance-classification examples $(d_0, l_0), (d_1, l_1), \cdots, (d_n, l_n)$. In a *multi-label* classification task, each instance is assigned a *subset* of labels $S \subseteq L$. A multi-label dataset D is therefore composed of n instance-classification examples $(d_0, S_0), (d_1, S_1), \cdots, (d_n, S_n)$. The multi-label problem is receiving increased attention and is relevant to many domains such as text classification [7, 4, 5], scene classification [8] and genomics [12, 10, 8].

All multi-label problems can be transformed into one or more single-label problems via some *problem transformation* (PT) [8]. In this fashion, any kind of single-label classifier can be used: single-label classifications are made and then transformed back into a multi-label representation. There are many reliable single-label classifiers, all of which can be employed under a PT method for multi-label classification. Some of the most successful PT approaches have worked with Support Vector Machines (SVMs) [4], Naive Bayes [6] and k Nearest Neighbor [12].

It is also possible to modify an existing single-label algorithm for the purpose of multi-label classification. Much of the literature is focussed on modifications to decision trees [10] and AdaBoost [7]. Essentially, these modifications simply employ some form of PT method internally and can often be generalised to any single-label classifier. Hence all solutions to multi-label classification involve some form of PT method.

There are essentially three fundamental PT methods[8]. They will be referred to in this paper as the Binary Method (BM), the Ranking Method (RM) and the Combination Method (CM).

The most widely used approach, the Binary Method (BM) [4, 12], learns $|L|$ binary classifiers $B_0, \cdots, B_{|L|}$. Each classifier B_j is responsible for predicting the $0/1$ association for each label $l_j \in L$.

Another commonly employed method, the Ranking Method (RM) [7], relies on a single-label classifier giving a probability distribution over all labels. The probabilities define a ranking for the labels. A threshold is used to determine the final subset of labels from this ranking.

Both BM and RM suffer from the *label independence* assumption, and fail to take advantage of any relationships between labels. This means they both may compose label sets whose elements would never co-occur in practise or unusually sized sets. Performance suffers accordingly.

The Combination Method (CM) [9] creates a single-label problem simply by treating each instance's label set S_i as an atomic label l'_i. For example, the multi-label set $\{a, c, d\}$ would become a single label acd. Hence the set of all distinct multi-label sets is transformed into a set of possible single labels L' to be considered by the single-label classifier.

CM overcomes the label independence problem, but suffers when labelling is very variable and many label combinations are unique or found infrequently in the dataset. This produces an overwhelming and imbalanced selection for the

1550-4786/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDM.2008.74

single-label classifier. A second crucial disadvantage is that CM can only classify examples with label sets found in the training set and thus new combinations cannot be formed.

An ensemble method for multi-label classification was recently pioneered by Tsoumakas and Vlahavas in a system called RAKEL (RAndom K-labEL subsets) [9]. For m iterations of the training data, RAKEL draws a random subset of size k from all labels L and trains a CM classifier using these labels. The authors use SVMs as the internal single-label classifier. A voting process using a threshold t determines the final classification set. Using appropriate values of m, k and t, RAKEL improves on BM and CM.

The following section presents the PS method. PS is a new method for multi-label classification that addresses the limitations of existing methods. It is designed to be fast and to feature low error rates over a wide range of multi-labelling scenarios. In later sections, PS is empirically evaluated and compared with the existing methods just described.

2 Pruning Sets

The motivation behind PS is to capitalise on the most important label relationships within a multi-label dataset. By pruning away infrequently occurring label sets, much unnecessary and detrimental complexity is avoided. A post-pruning step breaks up the pruned sets into more frequently occurring subsets, and is able to reintroduce pruned instances into the data, ensuring minimal information loss. The pruning operation is controlled by a parameter p which determines how often a label combination must occur for it *not* to be pruned. The PS method consists of the following phases:

Initialisation: D is the multi-label training set. A new empty training set D' is created to hold the final pruned training set. An empty set L' is also created to store label sets with counts of their occurrences in D.

Phase 1. Consider each label set S_i from each training example $(d_i, S_i) \in D$. If (S_i, c) can be found in L' for any count of c, then c is incremented by 1, otherwise a new pair $(S_i, 1)$ is added to L'.

Phase 2. The pruning parameter p is now considered. Pruning is done via exclusion from the set D'. Only training examples $(d_i, S_i) \in D$ where $(S_i, c) \in L'$ for $c > p$ are added directly to D'. The rejected (pruned) examples are passed on to Phase 3.

Phase 3. Training examples which were rejected by the pruning parameter at *Phase 2* can be reintroduced, along with information about their label relationships. This is done by decomposing each S_i (from each rejected example (d_i, S_i)) into subsets $s_{i0}, s_{i1}, \cdots, s_{in}$ where each $(s_{ij}, c) \in L'$ for $c > p$. These subsets are used to form *new* examples: $(d_i, s_{i0}), (d_i, s_{i1}), \cdots, (d_i, s_{in})$ which may then be added to D'. This is discussed below.

Phase 4. Finally a single-label representation is formed from D' using a training procedure like the one used for CM. This preserves the core label relationships in the form of combinations within data upon which any single-label classifier can be employed.

In *Phase 3*, pruned instances are reintroduced into the training in the form of new examples with smaller and more commonly found label sets. This preserves the example and information about its label set, however it is not beneficial to make new examples from every possible label subset. Aside from the obvious increased size of the training set, the average number of labels per instance becomes lower which can in turn cause too few labels to be predicted at classification time.

Hence a strategy is necessary to balance the trade off between preserving information and adding too many examples with smaller label sets. We present two such strategies for selecting label subsets to add. Each strategy has a parameter b. Recalling that in *Phase 3*, that for each $S_i \in D$ we generate every subset $s_{ij} \subset S_i$ where $(s_{ij}, c) \in L'$ and $c > p$. The strategies are (A): to rank these subsets firstly by the number of labels they contain and secondly by count c, then keep the *top* b ranked subsets — or — (B): to keep *all* subsets *of size greater than* b.

2.1 Ensembles of Pruned Sets

PS, as described so far, functions as a standalone method and in many domains improves over the other methods. However, it can not yet create new multi-label sets which have not been seen in the training data. This presents a problem when working with datasets where labelling is particularly irregular or complex. A general and flexible method is to combine the results of several classifiers in an ensemble.

We propose an ensemble of PS (EPS). PS is particularly suited to an ensemble due to its fast build times and, additionally, the ensemble counters any over-fitting effects of the pruning process and allows the creation of new label sets at classification time.

The build phase of EPS is straightforward. Over m iterations, a subset of the training set (we use 63%) is sampled and a PS classifier with relevant parameters is trained upon this subset (for a total of m classifiers).

The voting classification scheme, detailed in Figure 1, is unique to the multi-label domain. Under a threshold t, different multi-label predictions are combined into a final prediction. This final label set prediction may not have been

```
CLASSIFY(test instance $d$,classifiers $C_{0\cdots m}$,threshold $t$)
$v \leftarrow (0,0,0,\cdots,0)$ ▷ vector of size $|L|$
for $i \leftarrow 0$ to $m$
    do
        $w \leftarrow C_i$.classify($d$)
        $v \leftarrow w + v$
for $j \leftarrow 0$ to $|L|$
    do $Y[j] \leftarrow (v[j] > t)$ ? 1 : 0
return $Y$
```

Figure 1. The classification phase of EPS.

known to any of the individual PS models, allowing greater classification potential.

3 Experimental Evaluation

In this section the performance of PS is demonstrated in an empirical comparison against the three base problem transformation methods as well as the RAKEL algorithm mentioned in Section 1. First we will outline some multi-label evaluation measures, and present a collection of multi-label datasets. Then the experimental process is detailed, and the results presented and discussed.

3.1 Evaluation Measures

A multi-label classifier will produce a label subset $Y_i \subseteq L$ as a classification for an instance d_i, which can be compared to the true classification $S_i \subseteq L$ to evaluate performance.

Measuring accuracy by evaluating each label separately ($|L| \times |D|$ binary problems) can be overly lenient considering that usually almost all labels are irrelevant for any given example. On the other hand, evaluating accuracy based on the proportion of correctly labelled *examples* (where an example is correct *only* when its label set is an exact match) can be overly harsh.

We use the accuracy measure defined in [8]. Given a classified multi-label test set D:

$$Acc(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|S_i \cap Y_i|}{|S_i \cup Y_i|}$$

We also consider the F_1 measure common to information retrieval. Where p_i and r_i are the precision and recall of the predicted labels Y_i from the true labels S_i for each instance $d_i \in D$:

$$F_1(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2 \times p_i \times r_i}{p_i + r_i}$$

Table 1. A collection of multi-label datasets.

	$\lvert D\rvert$	$\lvert L\rvert$	$LCard(D)$	$PDist(D)$
Scene	2407	6	1.07	0.006
Medical	978	45	1.25	0.096
Yeast	2417	14	4.24	0.082
Enron	1702	53	3.38	0.442
Reuters	6000	103	1.46	0.147

We also use Hamming loss [8]; the symmetrical difference between Y_i and S_i averaged over all test examples:

$$Hloss(D) = 1 - \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|S_i \oplus Y_i|}{|L|}$$

3.2 Datasets

For these experiments we have collected a variety of datasets from different domains. Table 1 displays their associated statistics. *L*abel *Card*inality [8] is a standard measure of "multi-labelled-ness"; the average number of labels relevant to each instance; defined for a dataset D as:

$$LCard(D) = \frac{\sum_{i=1}^{|D|} |S_i|}{|D|}$$

We present also a measure for the *P*roportion of *Dist*inct label combinations. This measure quantifies the number of distinct label subsets relative to the total number of examples. It is useful for judging the complexity or "regularity" of a labelling scheme. For D:

$$PDist(D) = \frac{|\{S | \exists(d,S) \in D\}|}{|D|}$$

The *Medical* dataset [2] is composed of documents with a free-text summary of patient symptom histories and prognoses which are used to predict insurance codes. The *Yeast* data [9] relates to protein classification. *Scene* [9] relates to the classification of still scenes. *Enron* is a subset of the Enron email corpus [1] labelled by [3]. *Reuters* is a subset of the Reuters RCV1 dataset [5]. The text datasets (*Medical*, *Enron*, and *Reuters*) were all parsed into word frequency vectors that can be obtained by request from the authors.

3.3 Experimental Setup

All experiments presented in this paper were carried out using the WEKA [11] framework. In every case SVMs are employed as the single-label classifier. Each method is evaluated by 5×2 fold cross validation (CV) on each dataset.

For consistency, the number of iterations (m) is set to 10 for all ensemble methods. All other parameters are tuned on the training data using internal 5 fold CV, as are the thresholds. Parameters are tuned first and then thresholds secondly in the fashion described below.

EPS finds its optimal parameters using a standalone PS model. RAKEL, which needs a threshold to run, is given the initial value of 0.5 (as suggested by its authors) which it later adjusts as described below.

During tuning, the values of the parameters were sampled in order of the theoretical complexity they added to each algorithm. For example RAKEL's k parameter was incremented from 2 (the minimum value), whereas the p parameter for PS was decremented from 5 (higher values are not likely to improve accuracy). The internal CV for trialling each parameter value was aborted if it took longer than one hour.

As detailed in the RAKEL paper, increments of parameter values of k were 2 when $|L| > 14$, and 1 otherwise. The PS method requires a strategy parameter s, denoted by A_b for strategy A and B_b for strategy B. Values of $1, 2, 3$ for parameter b are examined in both cases.

Once parameter values have been selected, thresholds are adjusted. This is also done using $5\times$ CV but, in this case, each fold is tested in a two stage process: the first stage finds and assigns the best threshold t to the nearest 0.1, and the second stage finds and assigns the best value to the nearest 0.01 within the range $t \pm 0.05$. The average taken over the five folds produces the final value of t.

It is worth noting that the optimal parameters and thresholds chosen for all algorithms generally tended to be optimal, or close to optimal, within the range of values they were able to test.

All experiments were carried out on AMD Athlon(tm) 64 CPUs at 2 GHz with 1 gigabyte of memory.

3.4 Results

Tables 2, 3 and 4 show the full evaluation results including means and standard deviations averaged over all rounds. Arrows show significance according to a paired t-test against the CM method which is most relevant to [E]PS and RAKEL.

The most frequent parameter configurations and the average thresholds discovered by the tuning phases are presented in Table 6.

The average build times are displayed (in seconds) in Table 5. These times represent only the time taken to build the complete model for the test data only and *do not* include the internal parameter and threshold tuning.

In order to fully examine the complexity of the parameter ranges, all the methods were also timed on a 50/50 train/test split of the *Reuters* dataset. This dataset was chosen specifically due to its high $|D|$ and $|L|$. In this scenario [E]PS and RAKEL were left to try the full range of values for their respective p and k parameters. The methods either completed or ran out of memory (denoted by DNF). A range of results from this experiment is displayed in Table 7.

4 Discussion

Both PS and EPS improve consistently on the standard methods across all measures of evaluation. The improvement is most pronounced on the *Yeast* and *Enron* datasets which have a relatively high label cardinality and are therefore likely to contain more multi-label relationships.

As expected, PS performs best in an ensemble scheme (EPS), which allows the formation of new label sets and also helps prevent against over-fitting. In terms of F_1 measure (Table 3) EPS is statistically superior to CM on all datasets except *Medical* (where the difference is insignificant). It also performs better than state-of-the-art RAKEL.

Standalone PS still performs very competitively overall and the times in Table 5 indicate its advantages for fast classification, even when compared to the naive methods BM and RM.

An interesting feature of standalone PS is that it performs relatively better in terms of accuracy than in F_1 measure (although not always statistically significant). This is because PS always prunes away and divides up the most infrequently occurring label sets which also tend to contain the most labels. At classification time, this translates into high precision at the cost of recall and hence the sub-optimal F_1 statistic. However, in many real world scenarios, a consistent emphasis on precision and accuracy is more important than an optimum trade-off between precision and recall. This trend is avoided under an ensemble scheme where new combinations are formed and precision and recall can be governed by the threshold.

The complexity of RAKEL is one of its main disadvantages. Although in some cases its average *final* build times shown in Table 5 are less than those for EPS, this is misleading. Unlike EPS, which can tune parameters on a single model (of PS), RAKEL's full ensemble must be built to trial each parameter setting for each fold of internal CV. Parameter tuning for RAKEL is therefore much more expensive and is often terminated prematurely according to the conditions outlined in Section 3.3. In other words, it is computationally expensive and sometimes infeasible for multi-label methods like RAKEL to discover optimal values for their parameters.

In Table 7 RAKEL runs out of memory when $K = 62$ after taking about 6 hours when $K = 61$. PS completes with its most time-expensive p value (1) in about 4 minutes and takes only six times longer in EPS's ensemble scheme. RAKEL is computationally limited to a smaller parameter range and this explains its poor accuracy and F_1 measure on *Reuters*.

Table 2. Accuracy.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	71.81±1.22	58.28±0.92 ↘	71.72±0.98	71.93±1.08	73.80±0.95	71.58±0.89
Yeast	51.98±0.93	49.64±0.88 ↘	51.95±0.62	52.82±1.30	55.03±0.93 ↗	54.49±0.98 ↗
Medical	74.71±1.32	73.00±1.08	72.71±1.56	74.63±1.51	74.45±2.28	72.55±2.32
Enron	41.02±1.08	38.64±1.05	27.22±0.31 ↘	42.15±0.81	44.09±0.90 ↗	42.98±0.63
Reuters	49.17±0.67	31.91±0.76 ↘	49.08±0.59	49.83±0.59	49.80±0.59	31.80±0.29 ↘

↗, ↘ statistically significant improvement or degradation

Table 3. F_1 measure.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	0.729±0.01	0.671±0.01 ↘	0.724±0.01	0.730±0.01	0.752±0.01 ↗	0.735±0.01
Medical	0.767±0.01	0.791±0.01 ↗	0.743±0.01	0.766±0.02	0.764±0.02	0.784±0.01
Yeast	0.633±0.01	0.630±0.01	0.649±0.01 ↗	0.643±0.01	0.665±0.01 ↗	0.664±0.01 ↗
Enron	0.502±0.01	0.504±0.01	0.335±0.00 ↘	0.520±0.01	0.543±0.01 ↗	0.543±0.01 ↗
Reuters	0.482±0.01	0.421±0.01 ↘	0.485±0.00	0.496±0.00	0.499±0.01 ↗	0.418±0.00 ↘

↗, ↘ statistically significant improvement or degradation

Table 4. Hamming loss.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	0.096±0.004	0.111±0.003 ↘	0.095±0.003	0.095±0.004	0.090±0.003	0.098±0.004
Medical	0.012±0.001	0.011±0.000	0.013±0.001	0.012±0.001	0.013±0.001	0.012±0.001
Yeast	0.213±0.005	0.202±0.005 ↗	0.212±0.009	0.209±0.007	0.211±0.005	0.217±0.008
Enron	0.057±0.001	0.060±0.001	0.055±0.001 ↗	0.055±0.001	0.058±0.001	0.057±0.001
Reuters	0.013±0.000	0.011±0.000 ↗	0.012±0.000 ↗	0.012±0.001	0.013±0.001	0.012±0.000 ↗

↗, ↘ statistically significant improvement or degradation (*N.B. lower is better*)

Table 5. Build time.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	9.8	10.4	3.7	3.8	18.3	9.2
Medical	36.4	7.7	11.9	9.7	51.2	3.4
Yeast	187.8	11.1	34.0	29.8	172.6	64.8
Enron	1565.8	50.9	84.7	59.5	246.1	465.3
Reuters	1379.1	51.7	72.5	176.2	911.9	110.8

Table 6. Parameters and thresholds.

D	RM	PS	EPS	RAKEL
t	p,s	p,s,t	k,t	
Scene	0.30	4,A_2	4,A_2,0.37	4,0.30
Medical	0.10	1,A_2	1,A_2,0.30	8,0.19
Yeast	0.09	2,$B_{2.5}$	3,B_3,0.07	5,0.20
Enron	0.10	1,B_2	1,B_2,0.08	10,0.09
Reuters	0.10	1,A_3	1,A_3,0.21	16,0.06

N.B. BM and CM do not require parameters

Table 7. Build time (s) for *Reuters*.

CM	1379				
BM	123				
RM	505				
$p=$	5	4	3	2	1
PS	41	58	80	135	246
EPS	194	277	408	719	1,553
$k=$	2	25	50	61*	102
RAKEL	10	350	3,627	*22,337*	DNF

*$k = 61$ is the largest value to complete

Table 8. Approximation of memory use.

BM	$\lvert L\rvert \times \lvert D\rvert$
RM	$\lvert D\rvert \times LCard(D)$
CM	$\lvert D\rvert$
PS	$PF(D,p) + DF(D,s)$
EPS	$(PF(\subset D,p) + DF(\subset D,s)) \times m$
RAKEL	$\lvert D\rvert \times k \times m$

Although it may be argued that RAKEL would perform better with greater computing resources, under such a scenario EPS could also easily increase its number of iterations. Adding iterations adds at most linear complexity whereas, in Table 7, we clearly see that RAKEL's build time increases by a factor of approximately ten each time k is doubled.

As an aside, we further discover from the results that PS's strategy parameter s appears predictable. Table 6 shows that strategy A is selected consistently where $LCard(D)$ is low, and B when high (refer also to Table 1).

Although in theory the asymptotic complexity bounds of PS and EPS are not reduced over those of CM or RAKEL, the practical difference cannot be underestimated. Multi-labelled data invariably feature label distributions conducive to the efficient operation of PS, as multi-label schemes are consistently dominated by a small minority of core label relationships. This assumption can be made despite the exponential number of combinations which are *theoretically* possible with an increasing label set L. This explains why PS performs fast despite a theoretical worst-case performance similar to other methods.

Memory use is examined in Table 8. It is approximated by the number of instances generated during the transformation of a training set D with L possible labels (irrespective of any internal single-label classifier). All values are "hard" except the Pruning Function $PF(D,p)$ and Decomposition Function $DF(D,s)$ (corresponding to Phases 2 and 3 in Section 2) which depend on the distribution of the data in D (and the p and s parameters, respectively). It is guaranteed that $PF(D,p) < |D|$ and that $DF(D,s) < |D| \times LCard(D)$, and also that the complexity of PF is inversely proportional to the complexity of DF. Also, according to the argument concerning the use of PS in practise presented above, PS tends towards logarithmic complexity with respect to p. So we observe that PS is efficient in terms of memory as well as speed.

Hence the improvements PS offers are not simply incremental. The error reduction over other methods is often statistically significant, and its performance scales favourably across a range of multi-label datasets from different domains including large datasets with thousands of examples and with over a hundred labels.

5 Conclusions and Future Work

This paper introduced a new method for multi-label classification which uses a pruning procedure to focus on core relationships within multi-label sets. This procedure reduces the complexity and potential for error associated with dealing with a large number of infrequent sets.

While fully functional as a standalone method, PS is particularly suited to ensembles due to its fast operation and because the randomisation inherent to ensembles counteracts any over-fitting introduced by the pruning phase. Hence PS was also run within an ensemble scheme (EPS) for further reductions to the error rate.

Empirical statistical evaluation shows that the methods presented in this paper are often superior alternatives to other multi-label methods over a range of multi-labelled datasets. In many cases the improvements were statistically significant and build times were frequently and considerably reduced. The computational and memory complexity were analysed both practically and theoretically. The PS methods can be applied effectively and efficiently to many multi-label classification tasks including large and complex multi-label datasets.

References

[1] CALO project: Enron email dataset. URL: `http://www-2.cs.cmu.edu/~enron/`.

[2] Computational medical center: Medical NLP challenge. URL: `http://www.computationalmedicine.org/challenge/index.php`.

[3] UC Berkeley enron email analysis project: UC Berkeley enron email analysis. URL: `http://bailando.sims.berkeley.edu/enron_email.html`.

[4] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2004.

[5] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[6] A. K. McCallum. Multi-label text classification with a mixture model trained by EM. In *Association for the Advancement of Artificial Intelligence workshop on text learning*, 1999.

[7] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[8] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007.

[9] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, 2007.

[10] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[11] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

[12] M.-L. Zhang and Z.-H. Zhou. A k-nearest neighbor based algorithm for multi-label classification. volume 2, pages 718–721. The IEEE Computational Intelligence Society, 2005.