A Propositional Approach to Textual Case Indexing

Nirmalie Wiratunga[1], Rob Lothian[1], Sutanu Chakraborti[1], and Ivan Koychev[2]

[1] School of Computing,
The Robert Gordon University,
Aberdeen AB25 1HG, Scotland, UK
{nw, rml, sc}@comp.rgu.ac.uk
[2] Institute of Mathematics and Informatics,
Bulgarian Academy of Science,
Sofia - 1113, Bulgaria
ikoychev@math.bas.bg

Abstract. Problem solving with experiences that are recorded in text form requires a mapping from text to structured cases, so that case comparison can provide informed feedback for reasoning. One of the challenges is to acquire an indexing vocabulary to describe cases. We explore the use of machine learning and statistical techniques to automate aspects of this acquisition task. A propositional semantic indexing tool, PSI, which forms its indexing vocabulary from new features extracted as logical combinations of existing keywords, is presented. We propose that such logical combinations correspond more closely to natural concepts and are more transparent than linear combinations. Experiments show PSI-derived case representations to have superior retrieval performance to the original keyword-based representations. PSI also has comparable performance to Latent Semantic Indexing, a popular dimensionality reduction technique for text, which unlike PSI generates linear combinations of the original features.

1 Introduction

Discovery of new features is an important pre-processing step for textual data. This process is commonly referred to as feature extraction, to distinguish it from feature selection, where no new features are created [18]. Feature selection and feature extraction share the aim of forming better dimensions to represent the data. Historically, there has been more research work carried out in feature selection [9,20,16] than in extraction for text pre-processing applied to text retrieval and text classification tasks. However, combinations of features are better able to tackle the ambiguities in text (e.g. synonyms and polysemys) that often plague feature selection approaches. Typically, feature extraction approaches generate linear combinations of the original features. The strong focus on classification effectiveness alone has increasingly justified these approaches, even though their black-box nature is not ideal for user interaction. This argument applies even more strongly to combinations of features using algebraic or higher mathematical functions. When feature extraction is applied to tasks such as help desk systems, medical or law document management, email management or even Spam filtering, there is often a need for user interaction to guide retrieval or to support incremental query elaboration. The primary communication mode between system and user has the extracted

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 380–391, 2005.

features as vocabulary. Hence, these features should be transparent as well as providing good dimensions for classification.

The need for features that aid user interaction is particularly strong in the field of Case-Based Reasoning (CBR), where transparency is an important element during retrieval and reuse of solutions to similar, previously solved problems. This view is enforced by research presented at a mixed initiative CBR workshop [2]. The indexing vocabulary of a CBR system refers to the set of features that are used to describe past experiences to be represented as cases in the case base. Vocabulary acquisition is generally a demanding knowledge engineering task, even more so when experiences are captured in text form. Analysis of text typically begins by identifying keywords with which an indexing vocabulary is formed at the keyword level [14]. It is here that there is an obvious opportunity to apply feature extraction for index vocabulary acquisition with a view to learning transparent and effective textual case representations.

The focus of this paper is extraction of features to automate acquisition of index vocabulary for knowledge reuse. Techniques presented in this paper are suited for applications where past experiences are captured in free text form and are pre-classified according to the types of problems they solve. We present a Propositional Semantic Indexing (PSI) tool, which extracts interpretable features that are logical combinations of keywords. We propose that such logical combinations correspond more closely to natural concepts and are more transparent than linear combinations. PSI employs boosting combined with rule mining to encourage learning of non-overlapping (or orthogonal) sets of propositional clauses. A similarity metric is introduced so that textual cases can be compared based on similarity between extracted logical clauses. Interpretability of these logical constructs creates new avenues for user interaction and naturally leads to the discovery of knowledge. PSI's feature extraction approach is compared with the popular dimensionality reduction technique Latent Semantic Indexing (LSI), which uses singular value decomposition to extract orthogonal features that are linear combinations of keywords [7]. Case representations that employ PSI's logical expressions are more comprehensible to domain experts and end-users compared to LSI's linear keyword combinations. Ideally we wish to achieve this expressiveness without significant loss in retrieval effectiveness.

We first establish our terminology for feature selection and extraction, before describing how PSI extracts features as logical combinations. We then describe LSI, highlighting the problem of interpretability with linear combinations. Finally we show that PSI's approach achieves comparable retrieval performance yet remains expressive.

2 Feature Selection and Extraction

Consider the hypothetical example in Figure 1 where the task is to weed out Spam from legitimate email related to AI. To assist with future message filtering these messages must be mapped onto a set of cases before they can be reused. We will refer to the set of all labelled documents (cases) as $\mathcal{D}$. The keyword-vector representation is commonly used to represent a document d by considering the presence or absence of words [17]. Essentially the set of features are the set of words $\mathcal{W}$ (e.g. "conference", "intelligent"). Accordingly a document d is represented as a pair $(\mathbf{x}, y)$, where $\mathbf{x} = (x_1, \ldots, x_{|\mathcal{W}|})$ is

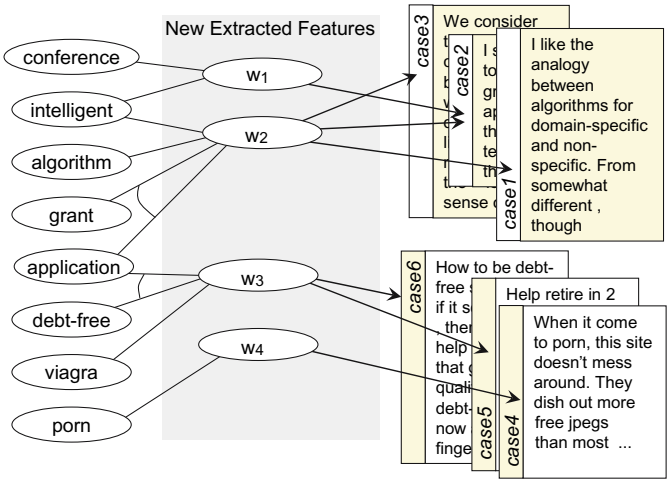

Fig. 1. Features as logical keyword combinations

a binary valued feature vector corresponding to the presence or absence of words in $\mathcal{W}$; and y is d's class label.

Feature selection reduces $|\mathcal{W}|$ to a smaller feature subset size m [20]. Information Gain (IG) is often used for this purpose, where m features with highest IG are retained and the new binary-valued feature vector $\mathbf{x}'$ is formed with the reduced word vocabulary set $\mathcal{W}'$, where $\mathcal{W}' \subset \mathcal{W}$ and $|\mathcal{W}'| \ll |\mathcal{W}|$. The new representation of document d with $\mathcal{W}'$ is a pair $(\mathbf{x}', y)$. Selection using IG is the base-line algorithm in this paper and is referred to as BASE. An obvious shortcoming of BASE is that it fails to ensure selection of non-redundant keywords. Ideally we want $\mathbf{x}'$ to contain features that are representative but also orthogonal. A more serious weakness is that BASE's one-to-one feature-word correspondence operates at a lexical level, ignoring underlying semantics.

Figure 1 illustrates a proof tree showing how new features can be extracted to capture keyword relationships using propositional disjunctive normal form clauses (DNF clauses). When keyword relationships are modelled, ambiguities in text can be resolved to some extent. For instance "grant" and "application" capture semantics akin to legitimate messages, while the same keyword "application" in conjunction with "debt-free" suggests Spam messages.

Feature extraction, like selection, also reduces $|\mathcal{W}|$ to a smaller feature subset size m. However unlike selected features, extracted features no longer correspond to presence or absence of single words. Therefore, with extracted features the new representation of document d is $(\mathbf{x}'', y)$, but $\mathcal{W}'' \not\subset \mathcal{W}$. When extracted features are logical combinations of keywords as in Figure 1, then a new feature $w'' \in \mathcal{W}''$, represents a propositional clause. For example the new feature w''_2 represents the clause: "intelligent" $\vee$ "algorithm" $\vee$ ("grant" $\wedge$ "application").

3 Propositional Semantic Indexing (PSI)

PSI discovers and captures underlying semantics in the form of propositional clauses. PSI's approach is two-fold. Firstly, decision stumps are selected by IG and refined by association rule mining, which discovers sets of Horn clause rules. Secondly, a boosting process encourages selection of non-redundant stumps. The PSI feature extraction algorithm and the instantiation of extracted features appear at the end of this section after a description of the main steps.

3.1 Decision Stump Guided Extraction

A decision stump is a one-level decision tree [12]. In PSI, a stump is initially formed using a single keyword, which is selected to maximise IG. An example decision stump formed with "conference" in its decision node appears in Figure 2. It partitions documents into leaves, based on whether or not "conference" appears in them. For instance 70 documents contain the word "conference" and just 5 of these are Spam (i.e. +5). It is not uncommon for documents containing "conference" to still be semantically similar to those not containing it. So documents containing "workshop" without "conference" in the right leaf are still contextually similar to those containing "conference" in the left leaf. A generalised decision node has the desired effect of bringing such semantically related documents closer [19]. Generalisation refines the decision node formed with a single feature w', to an extracted feature w'', containing a propositional clause. Typically a refined node results in an improved split (see right stump in Figure 2).

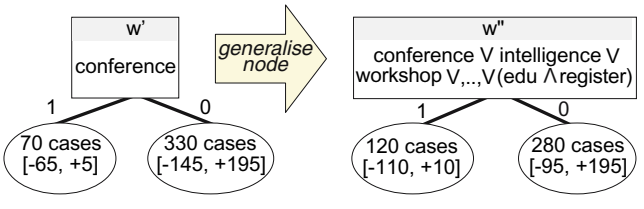

Fig. 2. Node Generalisation

A propositional clause is formed by adding disjuncts to an initial clause containing just the selected feature w'. Each disjunct is a conjunction of one or more keyword co-occurrences with similar contextual meaning to that of w'. An exhaustive search for disjuncts will invariably be impractical. Fortunately the search space can be pruned by using w' as a handle over this space. Instead of generating and evaluating all disjuncts, we generate propositional Horn clause rules that conclude w' given other logical keyword combinations.

3.2 Growing Clauses from Rules

Examples of five association rules concluding in "conference" (i.e. w') appear in Figure 3. These rules are of the form H $\leftarrow$ B, where the rule body B is a conjunction of

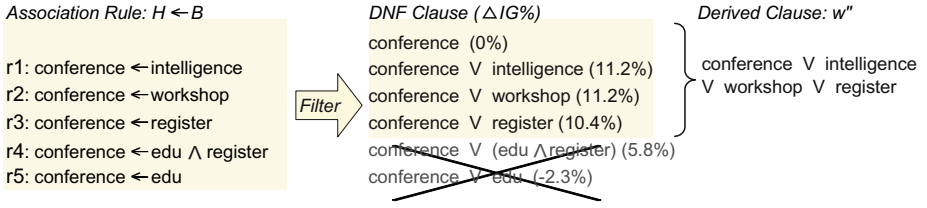

Fig. 3. Growing clauses from selected rules

keywords, and the rule head H is a single keyword. These conjunctions are keyword combinations that have been found to co-occur with the head keyword. Rule bodies are a good source of disjuncts with which to grow our DNF clause w'', which initially contains only the selected keyword "conference". However, an informed selection strategy is necessary to identify those disjuncts that are good descriptors of underlying semantics.

The contribution of each disjunct to clause growth is measured by comparing IG of w'' with and without the disjunct (rule body) included in the DNF clause. Disjuncts that fail to improve IG are filtered out by the *gain-filter*. Those remaining are passed onto *gen-filter* where any specialised forms of a disjunct with a lower IG compared to any one of its generalised forms are filtered out. The DNF clauses in Figure 3 show how each rule is converted into a potential DNF clause (difference in IG, used for filtering appear in brackets). The final DNF clause derived once the filtering step is completed is: "conference" $\vee$ "intelligence" $\vee$ "workshop" $\vee$ "register". We use the Apriori [1] association rule learner to generate feature extraction rules that conclude a selected w'. Apriori typically generates many rules, but the filters are able to identify useful rules.

3.3 Feature Extraction with Boosting

PSI's iterative approach to feature extraction employs boosted decision stumps (see Figure 4). The number of features to be extracted is determined by *vocabulary_size*. The general idea of boosting is to iteratively generate several (weak) learners, with each learner biased by the training error in the previous iteration [10]. This bias is expressed by modifying weights associated with documents. When boosted stumps are used for feature selection the new document distribution discourages selection of a redundant feature given the previously selected feature [6]. Here, with extracted features, unlike with single keyword-based features, we need to discourage discovery of an overlapping clause given the previously discovered clause. We achieve this by updating document weights in PSI according to the error of the decision stump created with the new extracted feature, w'', instead of w'.

3.4 Feature Instantiation

Once PSI has extracted new features, textual cases are mapped to a new representation. For a new feature w_i'', let $\mathcal{S}_i = \bigvee_j s_{ij}$, be its propositional clause, where $s_{ij} = \bigwedge_k x_{ijk}$

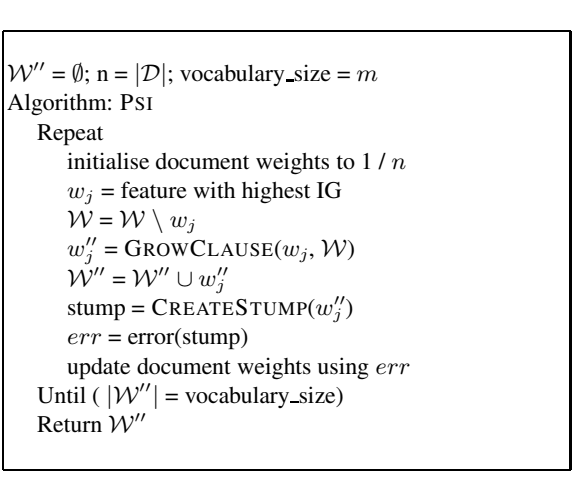

Fig. 4. Feature Extraction with PSI

is the jth conjunction in this clause. The new representation of document $d = (\mathbf{x}'', y)$ is obtained by:

$$x_i'' = \sum_j \text{gain\_inc}(s_{ij}) * \text{infer}(s_{ij})$$

here gain_inc returns the increase in gain achieved by s_{ij} when growing $\mathcal{S}_i$. Whether or not s_{ij} can be inferred (satisfied) from a document's initial representation $d = (\mathbf{x}, y)$ (i.e. using all features in $\mathcal{W}$) is determined by:

$$\text{infer}(s_{ij}) = \begin{cases} 1 & \text{if } (\bigwedge_k x_{ijk}) = \text{True} \\ 0 & \text{otherwise} \end{cases}$$

The PSI-derived representation enables case comparison at a semantic (or conceptual) level, because instantiated features now capture the degree to which each clause is satisfied by documents. In other words, satisfaction of the same disjunct will contribute more towards similarity than satisfaction of different disjuncts in the same clause.

4 Latent Semantic Indexing (LSI)

LSI is an established method of feature extraction and dimension reduction. The matrix whose columns are the document vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{|D|}$, known as the term-document matrix, constitutes a vector space representation of the document collection. In LSI, the term-document matrix is subjected to singular value decomposition (SVD). The SVD extracts an orthogonal basis for this space, consisting of new features that are linear combinations of the original features (keywords). Crucially, these new features are ranked according to their importance. It is assumed that the m highest-ranked features contain the true semantic structure of the document collection and the remaining features, which are considered to be noise, are discarded. Any value of m less than the

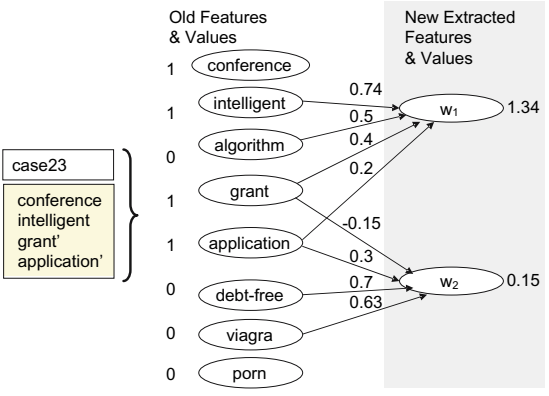

Fig. 5. Feature extraction using LSI

rank of the term-document matrix can be used, but good values will be much smaller than this rank and, hence, much smaller than either $|W|$ or $|D|$. These features form a new lower-dimensional representation which has frequently been found to improve performance in information retrieval and classification tasks [18,22]. Full technical details can be found in the original paper [7].

Figure 5 shows a hypothetical example from the AI-Spam domain (cf. Figure 1). The first extracted feature is a combination of "intelligent", "algorithm", "grant" and "application". Any document containing most of these is likely to be legitimate, so high values of this feature indicate non-Spam. The second feature has positive weights for "application", "debt-free" and "viagra" and a negative weight for "grant". A high value for this feature is likely to indicate Spam. The new features are orthogonal, despite having two keywords in common. The first feature has positive weights for both "grant" and "application", whereas the second has a negative weight for "grant". This shows how the modifying effect of "grant" on "application" might manifest itself in a LSI-derived representation. With a high score for the first extracted feature and a low score for the second, the incoming test case is likely to be classified as legitimate email.

LSI extracted features are linear combinations of typically very large numbers of keywords. In practice this can be in the order of hundreds/thousands of keywords, unlike in our illustrative example involving just 8 keywords. Consequently, it is difficult to interpret these features in a meaningful way. In contrast, a feature extracted by PSI combines far fewer keywords and its logical description of underlying semantics is easier to interpret. A further difference is that, although both PSI and LSI exploit word-word co-occurrences to discover and preserve underlying semantics, PSI also draws on word-class co-occurrences while LSI does not naturally exploit this information.

5 Evaluation

The goodness of case representations derived by BASE, LSI and PSI in terms of retrieval performance is compared on a retrieve-only CBR system, where the weighted majority

vote from the 3 best matching cases are re-used to classify the test case. A modified case similarity metric is used so that similarity due to absence of words (or words in linear combinations or in clauses) is treated as less important compared to their presence [19].

Experiments were conducted on 6 datasets; 4 involving email routing tasks and 2 involving Spam filtering. Various groups from the 20Newsgroups corpus of 20 Usenet groups [13], with 1000 postings (of discussions, queries, comments etc.) per group, form the routing datasets: SCIENCE (4 science related groups); REC (4 recreation related groups); HW (2 hardware problem discussion groups, one on Mac, the other on PC); and RELPOL (2 groups, one concerning religion, the other politics in the middle-east). Of the 2 Spam filtering datasets: USREMAIL [8] contains 1000 personal emails of which 50% are Spam; and LINGSPAM [16] contains 2893 messages from a linguistics mailing list of which 27% are Spam.

Equal-sized disjoint train-test splits were formed. Each split contains 20% of the dataset and also preserves the class distribution of the original corpus. All text was pre-processed by removing stop words (common words) and punctuation. Remaining words were stemmed to form $\mathcal{W}$, where $|\mathcal{W}|$ varies from approximately 1,000 in USREMAIL to 20,000 in LINGSPAM. Generally, with both routing and filtering tasks, the overall aim is to assign incoming messages into appropriate groups. Hence, test set accuracy was chosen as the primary measure of the effectiveness of the case representation as a facilitator of case comparison. For each test corpus and each method, the accuracy (averaged over 15 trials) was computed for representations with 20, 40, 60, 80, 100 and 120 features.

Paired t-tests were used to find improvements by LSI and PSI compared to BASE (one-tailed test) and differences between LSI and PSI (two-tailed test), both at the 95% significance level. Precision[1] is an important measure when comparing Spam filters, because it penalises error due to false positives (Legitimate → Spam). Hence, for the Spam filtering datasets, precision was tested as well as accuracy.

5.1 Results

Accuracy results in Figure 6 shows that BASE performs poorly with only 20 features, but gets closer to the superior PSI when more features are added. PSI's performance is normally good with 20 features and is robust to the feature subset size compared to both BASE and LSI. LSI clearly performs better for smaller sizes. This motivated an investigation of LSI with fewer than 20 features. We found that 10-feature LSI consistently outperforms 20-feature LSI and is close to optimal. Consequently, 10-feature LSI was used for the significance testing, in order to give a more realistic comparison with the other methods.

Table 5.1 compares performance of BASE and PSI (both 20 features) and LSI (10 features). Where LSI or PSI are significantly better than BASE, the results are in bold. Where LSI and PSI are significantly different, the better result is starred. It can be seen that LSI is significantly better than BASE on 6 of 8 measures and to PSI on 3. PSI is better than BASE on 7 measures and better than LSI on 2. We conclude that the 20-dimensional representations extracted by PSI have comparable effectiveness to the 10-dimensional representations extracted by LSI. Generally, BASE needs a much larger

[1] Precision = TP/(TP+FP) where TP is no. of true positives and FP is no. of false positives.

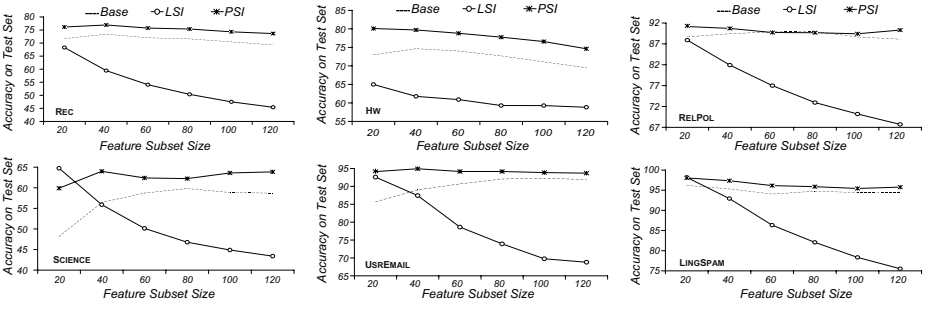

Fig. 6. Accuracy results for datasets

Table 1. Summary of significance testing for feature subset size 20 (10 for LSI)

	Routing: Accuracy				*Filtering: Accuracy (Precision)*	
Algo.	REC	HW	RELPOL	SCIENCE	USREMAIL	LINGSPAM
BASE	71.7	73.0	88.7	48.1	85.7 (89.5)	94.2 (92.0)
LSI	***78.7**	65.5	**90.4**	***71.8**	**93.9** (***96.8**)	**96.8** (89.0)
PSI	**76.2**	***80.1**	**91.2**	**59.9**	**94.1** (**95.2**)	**95.8** (*92.1)

indexing vocabulary to achieve comparable performance. PSI works well with a small vocabulary of features, which are more expressive than LSI's linear combinations.

5.2 Interpretability

Figure 7 provides a high-level view of sample features extracted by PSI in the form of logical combinations (for 3 of the datasets). It is interesting to compare the differences in extracted combinations (edges), the contribution of keywords (ovals) to different extracted features (boxes) and the number of keywords used to form conjunctions (usually not more than 3). We see a mass of interconnected nodes with the HW dataset on which PSI's performance was far superior to that of LSI. Closer examination of this data set shows that there are many keywords that are polysemous given the two classes. For instance "drive" is applicable both to Macs and PCs but combined with "vlb" indicates PC while with "syquest" indicates Mac. Unlike HW, the multi-class SCIENCE dataset contains several disjoint graphs each relating to a class, suggesting that these concepts are easily separable. Accuracy results show LSI to be a clear winner on SCIENCE. This further supports our observation that LSI operates best only in the absence of class-specific polysemous relationships. Finally, features extracted from LINGSPAM in figure 7 show that the majority of new features are single keywords rather than logical combinations. This explains BASE's good performance on LINGSPAM.

An obvious advantage of interpretability is knowledge discovery. Consider the SCIENCE tree, here, without the extracted clauses indicating that "msg", "food" and "chinese" are linked through "diet", one would not understand the meaning in context of a

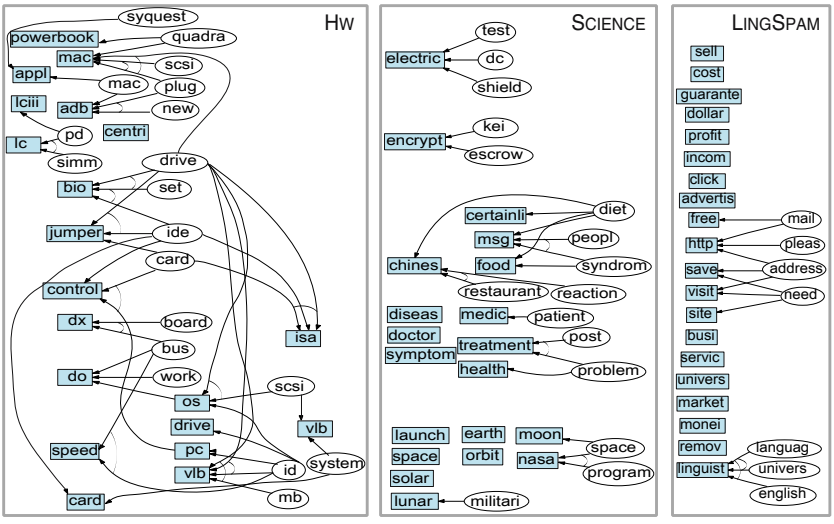

Fig. 7. Logical combinations extracted from datasets

term such as "msg". Such proof trees, which are automatically generated by PSI, highlight relations between keywords; this knowledge can further aid with glossary generation, often a demanding manual task (e.g. FALLQ [14]). Case authoring is a further task that can benefit from PSI generated trees. For example disjoint graphs involving "electric" and "encrypt" with many fewer keyword associations may suggest the need for case creation or discovery in that area. From a retrieval standpoint, PSI generated features can be exploited to facilitate query elaboration, within incremental retrieval systems.The main benefit to the user would be the ability to tailor the expanded query by deactivating disjuncts to suit retrieval needs. Since clauses are granular and can be broken down into semantically rich constituents, retrieval systems can gather statistics of which clauses worked well in the past, based on user interaction; this is difficult over linear combinations of features mined by LSI.

6 Related Work

Feature extraction is an important area of research particularly when dealing with textual data. In Textual-CBR (TCBR) research the SMILE system provides useful insight into how machine learning and statistical techniques can be employed to reason with legal documents [3]. As in PSI, single keywords in decision nodes are augmented with other keywords of similar context. Unlike PSI, these keywords are obtained by looking up a manually created domain-specific thesaurus. Although PSI grows clauses by analysing keyword co-occurrence patterns, they can just as easily be grown using existing domain-specific knowledge. A more recent interest in TCBR involves extraction of features in the form of predicates. The FACIT framework involving semi-automated index vocabulary acquisition addresses this challenge but also highlights the need for

reliance on deep syntactic parsing and the acquisition of a generative lexicon which warrants significant manual intervention [11].

In text classification and text mining research, there is much evidence to show that analysis of keyword relationships and modelling them as rules is a successful strategy for text retrieval. A good example is RIPPER [5], which adopts complex optimisation heuristics to learn propositional clauses for classification. A RIPPER rule is a Horn clause rule that concludes a class. In contrast, PSI's propositional clauses form features that can easily be exploited by CBR systems to enable case comparison at a semantic level. Such comparisons can also be facilitated with the FEATUREMINE [21] algorithm, which also employs association rule mining to create new features based on keyword co-occurrences. FEATUREMINE generates all possible pair-wise keyword co-occurrences converting only those that pass a significance test into new features. What is unique about PSI's approach is that firstly, search for associations is guided by an initial feature selection step, secondly, associations remaining after an informed filtering step are used to grow clauses, and, crucially, boosting is employed to encourage growing of non-overlapping clauses. The main advantage of PSI's approach is that instead of textual case similarity based solely on instantiated feature value comparisons (as in FEATUREMINE), PSI's clauses enable more fine-grained similarity comparisons. Like PSI, WHIRL [4] also integrates rules resulting in a more fine-grained similarity computation over text. However these rules are manually acquired.

The use of automated rule learning in an Information Extraction (IE) setting is demonstrated by TEXTRISE, where mined rules predict text for slots based on information extracted over other slots [15]. The vocabulary is thus limited to template slot fillers. In contrast PSI does not assume knowledge of case structures and is potentially more useful in unconstrained domains.

7 Conclusion

A novel contribution of this paper is the acquisition of an indexing vocabulary in the form of expressive clauses, and a case representation that captures the degree to which each clause is satisfied by documents. The propositional semantic indexing tool, PSI, introduced in the paper, enables text comparison at a semantic, instead of a lexical, level. Experiments show that PSI's retrieval performance is significantly better than that of retrieval over keyword-based representations. Comparison of PSI-derived representations with the popular LSI-derived representations generally shows comparable retrieval performance. However in the presence of class-specific polysemous relationships PSI is the clear winner. These results are very encouraging, because, although features extracted by LSI are rich mathematical descriptors of the underlying semantics in the domain, unlike PSI, they lack interpretability. We note that PSI's reliance on class knowledge inevitably restricts its range of applicability. Accordingly, future research will seek to develop an unsupervised version of PSI.

Acknowledgements

We thank Susan Craw for helpful discussions on this work.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in KD and DM*, pages 307–327, 1995. AAAI/MIT.
2. D. Aha, editor. *Mixed-Initiatives Workshop at 6th ECCBR*, 2002. Springer.
3. S. Bruninghaus and K. D. Ashley. Bootstrapping case base development with annotated case summaries. In *Proc of the 2nd ICCBR*, pages 59–73, 1999. Springer.
4. W. W. Cohen. Providing database-like access to the web using queries based on textual similarity. In *Proc of the Int Conf on Management of Data*, pages 558–560, 1998.
5. W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorisation. *ACM Transactions in Information Systems*, 17(2):141–173, 1999.
6. S. Das. Filters, wrappers and a boosting based hybrid for feature selection. In *Proc of the 18th ICML*, pages 74–81, 2001. Morgan Kaufmann.
7. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
8. S. J. Delany and P. Cunningham. An analysis of case-base editing in a spam filtering system. In *Proc of the 7th ECCBR*, pages 128–141, 2004. Springer.
9. G. Forman and I. Cohen. Learning with Little: Comparison of Classifiers Given Little Training. In *Proc of the 8th European Conf on PKDD*. pages 161–172, 2004.
10. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc of the 13th ICML*, pages 148–156, 1996.
11. K. M. Gupta and D. W. Aha. Towards acquiring case indexing taxonomies from text. In *Proc of the 17th Int FLAIRS Conference*, pages 307–315, 2004. AAAI press.
12. W. Iba and P. Langley. Induction of one-level decision trees. In *Proc of the 9th Int Workshop on Machine Learning*, pages 233–240, 1992.
13. T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF. Technical report, Carnegie Mellon University CMU-CS-96-118, 1996.
14. M. Lenz. Defining knowledge layers for textual CBR. In *Proc of the 4th European Workshop on CBR*, pages 298–309, 1998. Springer.
15. U. Y. Nahm and R. J. Mooney. Mining soft-matching rules from textual data. In *Proc of the 17th IJCAI*, pages 979–984, 2001.
16. G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6:49–73, 2003.
17. G. Salton and M. J. McGill. *An introduction to modern IR*. 1983, McGraw-Hill.
18. F. Sebastiani. ML in automated text categorisation. *ACM Computing surveys*, 34:1–47, 2002.
19. N. Wiratunga, I. Koychev, and S. Massie. Feature selection and generalisation for textual retrieval. In *Proc of the 7th ECCBR*, pages 806–820, 2004. Springer.
20. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorisation. In *Proc of the 14th ICML*, pages 412–420, 1997. Springer.
21. S. Zelikovitz. Mining for features to improve classification. In *Proc of Machine Learning, Models, Technologies and Applications*, 2003.
22. S. Zelikovitz and H. Hirsh. Using LSI for text classification in the presence of background text. In *Proc of the 10th Int Conf on Information and KM*, 2001.