Acquiring Word Similarities with Higher Order Association Mining

Sutanu Chakraborti, Nirmalie Wiratunga, Robert Lothian, and Stuart Watt

School of Computing,
The Robert Gordon University
Aberdeen AB25 1HG, Scotland, UK
{sc,rml,nw,sw}@comp.rgu.ac.uk

Abstract. We present a novel approach to mine word similarity in Textual Case Based Reasoning. We exploit indirect associations of words, in addition to direct ones for estimating their similarity. If word *A* co-occurs with word *B*, we say *A* and *B* share a first order association between them. If *A* co-occurs with *B* in some documents, and *B* with *C* in some others, then *A* and *C* are said to share a second order co-occurrence via *B*. Higher orders of co-occurrence may similarly be defined. In this paper we present algorithms for mining higher order co-occurrences. A weighted linear model is used to combine the contribution of these higher orders into a word similarity model. Our experimental results demonstrate significant improvements compared to similarity models based on first order co-occurrences alone. Our approach also outperforms state-of-the-art techniques like SVM and LSI in classification tasks of varying complexity.

1 Introduction

Textual Case Based Reasoning (TCBR) is based on the idea of modelling unstructured documents as cases. A knowledge light approach towards TCBR would use a bag of words directly to represent cases. The set of distinct terms and key-phrases in the document collection is treated as the feature set. One is tempted to believe that this line of thinking undermines the importance of domain-specific knowledge and thus blurs the distinction between CBR and Information Retrieval (IR) [2][3]. However, it may be argued that knowledge light approaches facilitate the application of statistical techniques to significantly lower knowledge acquisition overheads, in comparison to knowledge intensive techniques.

This paper presents a novel knowledge light technique for acquiring word similarities for TCBR. Our discussion is centred on a Case Retrieval Network (CRN) formalism, which has been demonstrated to be effective and efficient in retrieval over large and high dimensional case bases, typical with textual data [18]. CRNs have two main knowledge containers: knowledge about how words in a domain are related to each other (similarity knowledge); and knowledge about relatedness of words to cases (relevance knowledge). Typically statistical approaches model similarity between two words based on the number of documents in the corpus where these words co-occur.

R.O. Weber and M.M. Richter (Eds.): ICCBR 2007, LNAI 4626, pp. 61–76, 2007.

Notwithstanding significant amount of both philosophical and pragmatic debate on whether co-occurrence is a robust basis for semantic similarity [3], this simple approach works fairly well in the presence of large and representative collections [17]. Also, unlike domain-independent linguistic resources like WordNet or Roget's Thesaurus, this approach can be used for estimating domain specific word similarities. In this paper, we show that we can do even better. We incorporate the notion of higher-order co-occurrence into our model of word similarity. The basic idea is to use indirect associations between words, in addition to direct ones. For example if words *car* and *chassis* co-occur in one document, and words *automobile* and *chassis* in another, we can infer that *car* and *automobile* are related to each other, even if they do not co-occur in any document. Such a relation is called a second-order association. We can extend this to orders higher than two. Several interesting examples showing the importance of second order associations have been reported in studies on large corpora. Lund and Burgess [4] observe that near-synonyms like *road* and *street* fail to co-occur in their huge corpus. In a French corpus containing 24-million words from the daily newspaper *Le Monde* in 1999, Lemaire and Denhiere [5] found 131 occurrences of *internet*, 94 occurrences of *web*, but no co-occurrences at all. However, both words are strongly associated. Experiments [5] show that higher order co-occurrences can be exploited to infer "semantic relatedness" [19] between *road* and *street*, and between *web* and *internet*. Throughout this paper, we use the word "similarity" as a measure of semantic relatedness, as opposed to a rigid semantic relation (like synonymy or hyponymy).

This paper presents algorithms for mining higher order associations between words. The strengths of these associations are combined to yield an estimate of word similarity. One primary goal of this work is to evaluate the goodness of the learnt similarity knowledge. In addition, we show how our approach can be extended to incorporate class knowledge in supervised classification tasks. We compare our approach with state of the art text classifiers like Support Vector Machines (SVM) and k Nearest Neighbours (kNN) based on Latent Semantic Indexing (LSI). The comparison with LSI is particularly significant in the light of empirical evidence [6] that LSI implicitly exploits higher order co-occurrence relations between words to arrive at a reduced dimensional representation of words and documents. We make a comparative study to illustrate the advantages of explicitly capturing higher order associations, as opposed to doing so implicitly as in LSI.

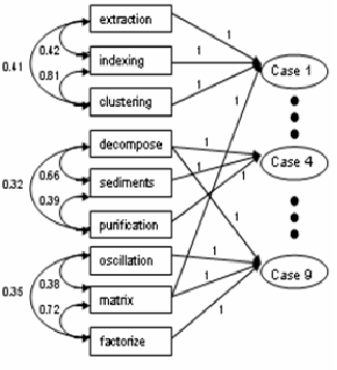
The rest of the paper is organized as follows. Section 2 introduces the CRN. Section 3 explains the concept of higher order associations, along with algorithms to mine the same. Section 4 describes our model of word similarities. Section 5 presents experimental findings comparing the performance of our model at the empirically determined best choice of parameters, with other approaches. All experiments reported in this paper were carried out on four text classification tasks of varied complexity. In Section 6, we present a novel approach of influencing the similarity values based on class knowledge, along with empirical results. Section 7 shows that the parameters of this model can be determined automatically. Possible extensions of the current work are discussed in Section 8. In Section 9, we situate our work in the context of other related work. Finally, section 10 summarizes our main contributions and concludes the paper.

2 Case Retrieval Networks

The CRN has been proposed as a representation formalism for CBR in [1]. To illustrate the basic idea we consider the example case-base in Fig. 1(a) which has nine cases comprising keywords, drawn from three domains: CBR, Chemistry and Linear Algebra. The keywords are along the columns of the matrix. Each case is represented as a row of binary values; a value 1 indicates that a keyword is present and 0 that it is absent. Cases 1, 2 and 3 relate to the CBR topic, cases 4, 5 and 6 to Chemistry and cases 7, 8 and 9 to Linear Algebra.

Document #	extraction	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	1	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	1	0	0	1	1	1

(a)



(b)

Fig. 1. CRN for Text Retrieval

Fig. 1(b) shows this case-base mapped onto a CRN. The keywords are treated as feature values, which are referred to as Information Entities (IEs). The rectangles denote IEs and the ovals represent cases. IE nodes are linked to case nodes by relevance arcs which are weighted according to the degree of association between terms and cases. In our example, relevance is 1 if the IE occurs in a case, 0 otherwise. The relevances are directly obtained from the matrix values in Fig. 1(a). IE nodes are related to each other by similarity arcs (circular arrows), which have numeric strengths denoting semantic similarity between two terms. For instance, the word "indexing" is more similar to "clustering" (similarity: 0.81) than to "extraction" (similarity: 0.42). Knowledge acquisition in the context of CRNs boils down to acquiring similarity and relevance values. This paper focuses on an approach to acquire similarity values automatically from a given collection of texts.

To perform retrieval, the query is parsed and IEs that appear in the query are activated. A similarity propagation is initiated through similarity arcs, to identify relevant IEs. The next step is relevance propagation, where the IEs in the query as well as those similar to the ones in the query spread activations to the case nodes via relevance arcs. These incoming activations are aggregated to form an activation score for each case node. Cases are ranked accordingly and the top *k* cases are retrieved.

A CRN facilitates efficient retrieval compared with a linear search through a case-base. While detailed time complexity estimates are available in [1], intuitively the speedup is because computation for establishing similarity between any distinct pair of IEs happens only once. Moreover, only cases with non-zero similarity to the query are taken into account in the retrieval process.

3 Higher Order Associations

The idea of higher order associations is illustrated through an example in Fig. 2. Terms *A* and *B* co-occur in Document 1 in Fig. 2(a), hence they are said to have a first order association between them. In Fig. 2(b), terms *A* and *C* co-occur in one document, and terms *C* and *B* in another. In our terminology, *A* and *B* share a second order association between them, through *C*. Extending this idea to Fig. 2(c), we say that *A* and *B* share a third order association between them through terms *C* and *D*. The similarity between two terms *A* and *B* is a function of the different orders of association between them. When modelled as a graph as shown in Fig. 2(d), each higher order association defines a path between the two vertices corresponding to terms *A* and *B*. (*A,C,B*) is a second order path and (*A,C,D,B*) is a third order path. An arc between any two nodes stands for a first-order co-occurrence relation between the corresponding words. A slightly more involved version is the weighted graph shown in Fig. 2(e). The weight of an arc connecting two nodes is proportional to the number of documents in the collection where they co-occur. It is important to note that while we have considered co-occurrence over entire documents, the context can be localized to arbitrary length word windows or sentences to restrict the number and scope of mined associations.

The basic idea is to estimate the strengths of different higher order co-occurrences and combine them into a word similarity model. Details of our similarity model appear in the next section. To estimate higher order strengths, we first tried a simple approach using goal driven unification supported by Prolog. The Prolog program has two parts to it: a fact base and a set of rules. The fact base was constructed automatically from the non-zero entries of the term document matrix, by taking all possible pairwise combinations of terms that appear in any document. From the matrix of Fig 1(a) we can construct facts such as

```
first_order(extraction, clustering).
first_order(extraction, matrix).
first_order(extraction, indexing).
```

Defining rules for higher order association is straightforward using Prolog. Second and third order associations are defined in the following statements:

```
second_order(X, Y ,Z) :- first_order(X, Z), first_order(Z, Y), X \== Y.
third_order(X,Y,Z,W) :- second_order(X,W,Z), first_order(W,Y), X \== Y,
Z\== Y.
```

Often, we are not interested in the actual words that act as links between words, as extracted by the Prolog unifications, but more in the *number* of distinct paths linking up words. This is easy in Prolog, as well:

```
lengthOfList([], 0).
lengthOfList ([_|Tail], N) :- lengthOfList (Tail, N1), N is 1 + N1.
no_of_2ord_paths(X,Y,N, List) :- setof(Z, second_order(X,Y,Z), List),
lengthOfList(List,N).
no_of_3ord_paths(X,Y,N, List1) :- setof((K,L), third_order(X,Y,K,L),
List1), lengthOfList(List1,N).
```

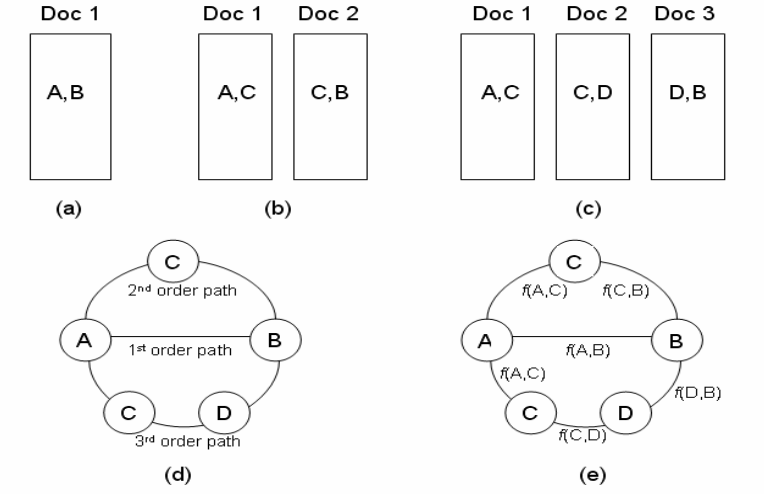


Fig. 2. Graphical Representation of Higher Order Co-occurrences

One main limitation of Prolog in this task is the combinatorial explosion in the number of first order associations that had to be recorded in the fact-base. In realistic tasks over several hundreds of documents, our version of Prolog (SWI-Prolog) often ran out of memory. To address this limitation, we explored the applicability of matrix operations to directly compute the strengths of higher order associations. We first implemented an approach reported by [7], where the authors start by computing a first order co-occurrence matrix. For |W| words in the feature set, this is a |W| × |W| matrix which has a value 1 in the *i,j*th element if word *i* co-occurs with word *j* in at least one document. For all pairs of words that do not co-occur in any document, the corresponding element in the matrix is 0. The diagonal values are set to zero since we are not interested in trivial co-occurrence of a word with itself. The first-order co-occurrence matrix is calculated using the following steps:

Step 1: The term document matrix A is multiplied with its transpose A^T to obtain the |W| × |W| matrix T_0.
Step 2: All non-zero values of T_0 are set to 1, and the diagonal values are set to zero to yield a binary first order co-occurrence matrix T.
Step 3: The second order co-occurrence matrix T_2 can be calculated by squaring T. The third order matrix T_3 is given as T^3. Other higher order co-occurrence matrices can be calculated similarly.

Before a matrix is reduced to binary, the value of its *i,j*th element is the number of co-occurrence paths between words *i* and *j*. The strength of a first order co-occurrence path is the number of documents in which two words co-occur. The strength of a second order co-occurrence path between words *a* and *b* is the number of distinct words *c* such that *a* co-occurs with *c* and *b* co-occurs with *c*.

Implementing the above algorithm revealed a critical shortcoming. Let us consider a third order association between terms *a* and *b* via terms *c* and *d*. Thus pairs *a* and *c*, *c* and *d*, and *d* and *b* co-occur with each other. In finding distinct pairs of terms *c* and *d*, we need to ensure that they are not the same as either *a* or *b*. By setting the diagonal elements to 0 in Step 2 above, the algorithm ensures that *a* and *c* are different, and so are *d* and *b*. But in addition we also need to ensure that *d* is not the same as *a*, and *c* is not the same as *b*, and this is not taken care of. Thus the strengths of third order associations were over-estimated by the algorithm. We need to make a correction to the algorithm to address this limitation. The brute force approach of explicitly counting terms that satisfy the above-mentioned constraint instead of blindly cubing the binary matrix *T*, turned out to be computationally expensive. We present below a technique that rewrites this procedure as an equivalent matrix manipulation, which can be implemented efficiently in matrix processing environments like Matlab.

Let *T* be the matrix of first order connections with diagonal elements set to zero. For third-order co-occurrences, we seek to enumerate paths of type *i-j-k-l* for all *i* and *l*. Now

$$\left(T^3\right)_{il} = \sum_{j,k} T_{ij} T_{jk} T_{kl}$$

is the total number of such paths, including paths of type *i-j-i-l* and *i-l-k-l*, which we wish to exclude. Let n_i be the number of paths of type *i-j-i*. This is equal to the total number of paths originating from *i*. We may evaluate n_i by summing the rows (or columns) of *T*:

$$n_i = \sum_j T_{ij}$$

Now, the number of paths of type *i-j-i-l* is $n_i T_{il}$ and for type *i-l-k-l* the count is $n_l T_{il}$. If $T_{il} \neq 0$, then we have counted the path *i-j-i-j* twice, so the total number of invalid paths is $(n_i+n_l-1)T_{il}$. Equivalently, if we construct a discount matrix *D* whose elements $D_{il} = (n_i+n_l-1)$, then the number of invalid paths between words *i* and *j* is given by the *i,j* th element of the pointwise product *D*T*. We use the following procedure:

(1) Calculate T^3.

(2) Enumerate and discount the invalid paths as above. T^3- *D*T* is the revised third order matrix.

3.1 An Example

We illustrate the above ideas on a toy case base comprising 4 terms and 4 documents as shown in Fig.3. The third order matrix T_3' says that there are two third-order paths between terms *t2* and *t3*, one third order path between *t1* and *t2*, another between terms *t1* and *t3*, and none between *t1* and *t4*. A closer inspection of matrix *T* reveals

that that this is indeed true. Fig. 4 shows a graphical representation of matrix *T*, where an arc exists between any two nodes iff the corresponding entry in the matrix is 1, denoting that there is at least one document in the collection that has both of these terms. The two third order paths between *t2* and *t3* are *t2-t1-t4-t3* and *t2-t4-t1-t3*. The only third order path between *t1* and *t2* is *t1-t3-t4-t2*, and between *t1* and *t3* is *t1-t2-t4-t3*. There are only two possible candidates for a third order path between *t1* and *t4*: *t1-t2-t3-t4* and *t1-t3-t2-t4*. Either would require a first order association between *t2* and *t3*, which in our example does not exist, since there are no documents that contain both *t2* and *t3*. Hence any third order association between *t1* and *t4* is ruled out.

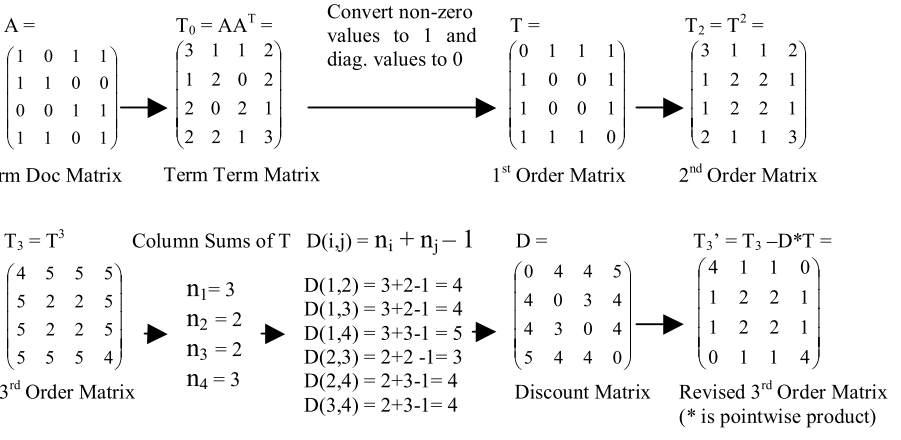


Fig. 3. An Example

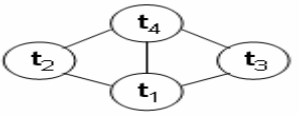


Fig. 4. The Term-Term Association Graph

4 Modeling Word Similarities

Once higher order co-occurrences are mined, we need to translate them into a measure of similarity between words. Intuition suggests that very high order co-occurrences do not really indicate similarity. In a study of higher order associations in the context of LSI [5], the authors report experimental evidence to confirm that associations beyond an order of 3 have a very weak influence on similarity modeled by LSI. In our word similarity model, we ignore the effects of orders higher than 3. In the last section, we have defined the strength of a higher order association between two terms as the number of co-occurrence paths between those terms. Let *first_order(a,b)*, *second_order(a,b)* and *third_order(a,b)* denote the strengths of first, second and third order associations between terms *a* and *b* respectively. The similarity between terms *a*

and *b* can be expressed as a weighted linear combination of the strengths of the first three orders of co-occurrence as follows:

$$similarity(a,b) = \alpha\, first\_order(a,b) + \beta\, second\_order(a,b) + \gamma\, third\_order(a,b) \quad (1)$$

Note that higher the order of association, the larger the number of co-occurrence paths (since $T^n_{i,j} > T^m_{i,j}$, if $n>m$ and if for all $T_{i,j} \neq 0$, $T_{i,j} \geq 1$, which is true in our case), and hence the greater the strength of association. Thus, to make α, β and γ comparable to each other, we need to normalize *first_order(a,b)*, *second_order(a,b)* and *third_order(a,b)* to values in [0,1]. In our implementation, we achieve this by dividing each of these values by the maximum value between any pair of words corresponding to that order. Each distinct choice of α, β and γ leads to a different set of similarities between terms, which can then be used as similarity arcs in the CRN to perform retrieval or classification. In complex domains, we would expect higher order associations to play a critical role and hence such domains should show preference for higher values of β and γ compared to simpler ones.

5 Experimental Results

Our first experiment has two goals. Firstly, we test the hypothesis that higher order co-occurrences indeed lead to better classification effectiveness. Secondly, we study the values of α, β and γ that lead to best performances in four classification tasks of varying complexity. These experiments are carried out by varying α, β and γ, computing term similarities at each of these settings as given by (1) above, and observing the classification accuracies achieved by the CRN with these similarity values.

5.1 Experimental Methodology

Experiments were conducted on four datasets, of which two involve text classification in routing tasks and two involve Spam filtering. It may be noted that while the results reported are based on classification tasks for ease of evaluation, the techniques presented in this paper are fairly general and can easily be adapted for unsupervised retrieval tasks as well.

The two datasets used for text classification in routing were formed from the 20 Newsgroups [8] corpus which has about 20,000 Usenet news postings organized into 20 different newsgroups. One thousand messages (of discussions, queries, comments etc.) from each of the twenty newsgroups were chosen at random and partitioned by the newsgroup name. We form the following two subcorpuses:

- HARDWARE which has 2 hardware problem discussion groups, one on Apple Mac and the other on PC
- RELPOL which has two groups, one concerning religion, the other politics

The two datasets used for evaluating performance on Spam filtering include

- USREMAIL contains 1000 personal emails of which 50% are spam
- LINGSPAM dataset which contains 2893 email messages, of which 83% are non-spam messages related to linguistics, the rest are spam

We created equal sized disjoint training and test sets, where each set contains 20 % of the dataset of documents randomly selected from the original corpus, preserving the class distribution of the original corpus. For repeated trials, 15 such train test splits were formed.

Textual cases were formed by pre-processing documents by removing stopwords (common words) and special characters such as quote marks, commas and full stops. Some special characters like "!", "@", "%" and "$" were retained since they have been found to be discriminative for some domains. Remaining words are reduced to their stem using the Porter's algorithm [9]. The word stems that remain after preprocessing constitute the set of IEs.

In our experiments, we took into account first, second and third order associations, as given by (1). α is set to 1, and β and γ are incremented steps of 0.1 in the range [0,1.9] to examine the effect of second and third orders. $\beta = 0$, $\gamma = 0$ corresponds to the situation where only first order associations are used. At each unique choice of the three parameters, the term–term similarities obtained with those settings are used to define the similarity arcs in a CRN. The relevance arcs were set to 1 or 0 based on whether an IE(word) is present or absent in a case. The CRN produces the dot product of the incoming case with each of the existing cases. These values are normalized using the query and case norms to obtain the cosine similarity. A weighted 3-nearest neighbour algorithm is used to classify the test document.

We compare the classification accuracies with two other classifiers. The first is Support Vector Machines (SVM) which is reported to yield state-of-the-art performance in text categorization. The second is Latent Semantic Indexing (LSI), which maps terms and documents to a lower dimensional "concept" space, which is hypothesized to be more robust to variations due to word choice. Cases are represented using the reduced dimensions obtained with LSI, and a usual *k*-NN approach can then be used for retrieval. The comparison of our approach with LSI is motivated by the observations in [6], which attribute LSI performance to its ability to implicitly model higher order associations between words. However unlike our approach, LSI is constrained by the need to maximize variance across the concept dimensions, and by the need to produce the best k-rank approximation to the original term document matrix, in the least-squares sense. Our intuition was that these constraints are unnecessarily restrictive in a classification domain and could be relaxed to obtain better performance. Unlike LSI, our approach *explicitly* captures higher order associations and embeds this into term-term similarity knowledge. This also opens avenues for better visualization as discussed in Section 8.

LSI performance is critically dependent on the number of concept dimensions used for representing terms and documents. To make a fair comparison, we report LSI performance at the dimension at which its performance was found to be optimal. For SVM, we used a linear kernel as this was reported to yield best results in text categorization tasks [10].

5.2 Analysis of Results

Table 2 presents a summary of the results. The figures in bold are the best results after paired t-tests between each classifier over results from the 15 trials. In situations where the differences between the top ranking classifiers is not statistically significant

($p > 0.05$), all top figures have been marked in bold. We observe that using second and third order co-occurrences at parameter settings that yield best performance results in better classification accuracies compared to using first-order co-occurrences alone ($\beta,\gamma = 0$). While the differences are statistically significant on all four datasets, the magnitude of improvement is more conspicuous in HARDWARE and RELPOL, which are harder domains, compared to USREMAIL and LINGSPAM, which already recorded high accuracies with simpler approaches. In the RELPOL domain, 16 terms provided second order path between *Bible* and *sin*; interestingly these include *Christ*, *Jesus*, *faith, scripture, heaven, roman, kill, genocide* and *biblical.* It may be noted that the use of higher order co-occurrences leads to better accuracies compared to LSI and the differences are statistically significant on all four domains. This is all the more noteworthy in the light of our paired tests that reveal that LSI does better than first order co-occurrences on both HARDWARE and RELPOL, while results are statistically equivalent on the other two datasets. These two observations show LSI does better than using first order associations alone, but is outperformed comprehendsively when higher orders are used.

We also note that our approach outperforms SVM on all datasets except HARDWARE where SVM performs significantly better. One possible reason for the relatively poor performance in HARDWARE could be a significant overlap in vocabularies used to describe problems in Mac and PC. The problem is compounded by the fact that we ignore class knowledge of training documents while constructing similarity relations between terms. In contrast this is a critical input to SVM. Motivated by this observation, we investigated a novel way of introducing class knowledge into the higher order mining algorithm, which is described in Section 6.

Table 1 reports α, β and γ values at which best performances are observed. Easier domains like USREMAIL and LINGSPAM appear to prefer lower values of β and γ compared to HARDWARE and RELPOL. We will re-examine this observation in the light of more experimental results in Section 7.

Table 1. Empirically determined best values of α,β and γ

	HARDWARE	**RELPOL**	**USREMAIL**	**LINGSPAM**
$(\alpha,\beta,\gamma)_{optimal}$	(1,0.37 ,1.15)	(1,0.61 ,1.04	(1,0.21,0.15)	(1,0.27, 0.31)

Table 2. Comparing classifier accuracies

	HARDWARE	**RELPOL**	**USREMAIL**	**LINGSPAM**
BASE(VSM) (Euclidean)	.5951	.7054	.5923	.8509
LSI-mined Similarities	.7240	.9339	.9583	.9832
SVM	**.7883**	.9228	**.9583**	.9636
First Order Similarities	.7171	.9309	.9577	.9826
Higher Order Similarities	.7451	**.9530**	**.9640**	**.9859**

6 Incorporating Class Knowledge into Word Similarities

In a supervised classification context, we have class knowledge of training documents in addition to the co-occurrence knowledge. Our intention is to incorporate this class knowledge as part of pre-processing. The idea is very similar to the approach described in [11], where LSI was extended to supervised classification tasks. Each document in the training set is padded with additional artificial terms that are representative of class knowledge. For example in the Hardware domain, all documents belonging to Apple Mac are augmented with artificial terms A, B , C and D, and all documents belonging to PC are padded with E, F, G and H. The padded terms, which we refer to as *sprinkled* terms, appear as new IEs in the CRN and are treated like any existing IE node. The revised architecture is shown in Fig. 5. When co-occurrences are mined on this new representation, terms representative of the same class are drawn closer to each other, and terms from disjoint classes are drawn farther apart. This happens because the sprinkled terms provide second-order co-occurrence paths between terms of the same class. For the test documents, the class is unknown; hence none of the artificial terms are activated. One important question is to decide the number of additional terms to be added for each class; an empirical solution is to use as many as yields best results over a cross validation dataset. While sprinkled terms help in emphasizing class knowledge, using too many of them may distort finer word association patterns in the original data [11]. In our experiments, we used 8 additional terms per class, as this was empirically found to yield good results.

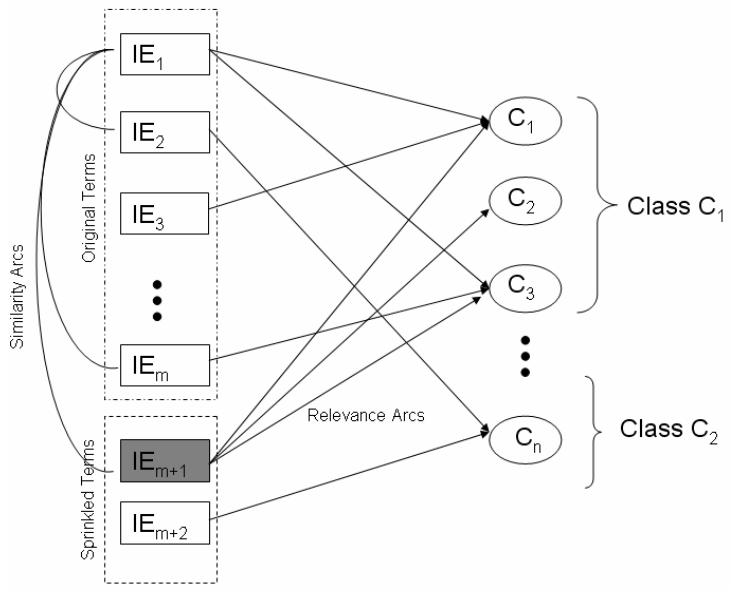


Fig. 5. A CRN Architecture after Sprinkling Terms that carry class knowledge

6.1 Empirical Results

The results are summarized in Tables 3 and 4. Sprinkling led to conspicuous improvement in performance over the HARDWARE dataset from 74.51% to 80.44%. This unambiguously points to the importance of class knowledge in this dataset. Table 3 suggests that sprinkled higher orders outperforms SVM on all datasets; in the USREMAIL dataset, the improvement is not statistically significant. This is possibly because the domain is simple and had already high recorded accuracies. For the RELPOL domain however, adding class knowledge led to a slight drop in the performance from 95.30% to 93.93% (Table 4), which was still significantly better than both LSI and SVM. The drop in RELPOL performance indicates that in this domain, class knowledge is not as important as in HARDWARE. In our current implementation, we have used uniform number of sprinkled terms over all domains. Performance could be improved by optimising the number of sprinkled terms for each individual domain. For example, HARDWARE would be more heavily sprinkled than RELPOL.

Table 3. Comparing Sprinkled Higher Orders against SVM

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
Sprinkled HO	**.8044**	**.9393**	**.9630**	**.9838**
SVM	.7883	.9228	**.9583**	.9636

Table 4. Comparing Higher Orders with and without Sprinkling

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
Sprinkled HO	**.8044**	.9393	**.9630**	**.9838**
Higher Order	.7451	**.9530**	**.9640**	**.9859**

7 Learning Model Parameters Automatically

Performing exhaustive search on the parameter space allows us to empirically ascertain the contributions of each co-occurrence order. However, in practice, we would need a mechanism to determine the parameters automatically based on a given text collection. We have investigated a Genetic Algorithm based approach to achieve this in supervised classification tasks. The parameters are learnt on the training set, with the objective of maximizing classification accuracy on the unseen test set. Since the test set is not available, we instead set our objective to optimizing classification accuracy over 5-fold cross validation on the training set. While details of our approach can be found in [21], we summarize our main findings below.

Table 5 presents the classification accuracies when the parameters were learnt using the GA-based approach. We used the architecture of Fig. 5 where sprinkled terms were used as carriers of class knowledge. The accuracy figures with the learnt

Table 5. Comparing effectiveness of empirically determined and GA-learnt parameters

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
Sprinkled HO (parameter learning)	.7938	.9304	**.9593**	.9814
Sprinkled HO	**.8044**	**.9393**	**.9630**	**.9838**

Table 6. Parameter values learnt by GA

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
$(\alpha,\beta,\gamma)_{optimal}$	(1,1.88 ,1.56)	(1,1.01 ,1.15)	(1,0.97,0.85)	(1,0.73, 0.96)

parameters are very similar to the figures obtained by the approach of Section 6 where the best values are chosen after exhaustively searching the parameter space in fixed increments. While there is still a statistically significant difference in three of four datasets, the very close average values suggest that the GA-based approach holds promise in significantly lowering manual overheads in parameter setting, while still continuing to deliver good performance. We need further research into better tuning of our approach for facilitating faster and more effective search in the parameter space. Table 6 shows the values of α,β and γ that were learnt by our algorithm for each of the four datasets. Comparing these values with the corresponding ones in Table 1, we observe a significant increase in the values of β. This can be attributed to the fact that sprinkled terms provide second order co-occurrence paths between terms of the same class. Increasing β thus helps in boosting similarity between terms of the same class, and decreasing similarity between terms of disjoint classes. This explains the greatly improved performance in the HARDWARE domain with sprinkling.

8 Discussion

While we have evaluated our ideas in the context of classification domains, it would be possible to apply the basic idea to unsupervised retrieval scenarios as well. One interesting metric to evaluate goodness of a TCBR configuration in unsupervised domains was recently proposed by Luc Lamontagne [12]. The measure, which the author calls case cohesion, measures the degree of correspondence between problem and solution components of textual cases. Using case cohesion instead of classification accuracy as a measure of the fitness function in our optimization algorithm would be a first cut towards applying our approach to retrieval tasks.

The importance of modeling similarity using higher order co-occurrences extends beyond textual CBR. In the context of recommender systems, several authors have reported problems due to sparseness of user-item matrices [16]; Semeraro et al [15] for example, report that 87% of the entries in their user-item matrix are zero. Knowledge representations used in collaborative recommenders (like concept lattices [24]) fail to exploit associations beyond the first order. Higher order associations can help reduce the sparseness and allow for better recommendation. In this context, analysis of higher-order associations in user item matrices will help discover novel

product recommendation rules that would normally be implicit in the user ratings. Our approach can also be applied to link analysis in social networks [23], for clustering similar words, and resolving ambiguity of words spanning several clusters.

While our approach has outperformed SVM, the important thing to note is the explicit nature of our similarity relations as compared to SVM. It is not clear how SVM can be used to mine similarity between words, or incorporate expert feedback. The comparison with SVM illustrates that our techniques can outperform the best-in-line classifier while being able to explicitize its knowledge content, and supporting lazy incremental updates, both of which are strengths of CBR. The Prolog-based system described in Section 3 has its own advantages for visualization. For any given pair of words, all higher order associations can be depicted in graphs of the kind shown in Fig.2, which may be useful for explanation or for initiating expert feedback.

9 Related Works

Several works in the past have pointed to the importance of higher order co-occurrence in modeling word similarity. However we have not come across any work that explicitly attempts to obtain a parameterized model of similarity based on these co-occurrences, and learn optimal values of these parameters based on a fitness criterion. The work by Kontostathis and Pottenger [6] provides empirical evidence to show that LSI implicitly exploits higher order co-occurrence paths between words to arrive at its revised representations. This provides a fresh explanation for improvements obtained using LSI in text retrieval applications. Edmonds [13] examines the role of higher order co-occurrence in addressing the problem of lexical choice, which is important to both machine translation and natural language generation. Broadly speaking, the goal is to determine which of the possible synonyms is most appropriate for a given communication (or pragmatic) goal. The authors show that using second order co-occurrence has a favourable influence on the performance of their lexical choice program. Recent work by Lemaire and Denhiere [5] makes an in-depth study of the relationship between similarity and co-occurrence in a huge corpus of children's texts. They show that while semantic similarity is largely associated with first order co-occurrence, the latter overestimates the former. Higher order co-occurrences as well as lone occurrences (occurrence of word *a* but not *b* and vice versa) were used to account for LSI-inferred term similarities. Unlike our work, the authors do not propose an algorithm to arrive at word similarities; their approach is more analytic than synthetic. Two other recent approaches potentially useful for mining word similarities are distributional word clustering for textual case indexing [20][22], and Propositional Semantic Indexing [2] which mines word relationships using Association Rule Mining (ARM) with the goal of feature generalization. However, probability estimates used in the first approach and the ARM approach used in the second currently fail to accommodate associations beyond the first order. It appears that both approaches can potentially benefit from higher-order knowledge.

10 Conclusion

The main contribution of this paper is an approach for exploiting higher-order associations between words to acquire similarity knowledge for CRNs. We demonstrated the importance of higher order co-occurrences in determining word similarity, presented both supervised and unsupervised algorithms for mining such associations and proposed a word similarity model, whose parameters are learnt using an evolutionary approach. We have demonstrated the effectiveness of the learnt similarity knowledge and shown that using second and third order-co-occurrences yields better results than using first-order co-occurrence alone. Another contribution of the current work is to incorporate class knowledge into the process of mining higher order associations. We have demonstrated the effectiveness of this extension as our approach outperforms state of the art classifiers like SVM and LSI/*k*NN on classification tasks of varying complexity. Though the work has been presented in the context of CRNs, in essence we have presented a general approach to mine feature similarities, which can be easily integrated into other retrieval formalisms. Future work will aim at improving the parameter learning algorithm, and forming an easy-to-use workbench for similarity knowledge mining, for textual and non-textual CBR applications.

References

1. Lenz, M., Burkhard, H.: Case Retrieval Nets: Foundations, Properties, Implementation, and Results, Technical Report, Humboldt-Universität zu Berlin (1996)
2. Wiratunga, N., Lothian, R., Chakraborti, S., Koychev, I.: A Propositional Approach to Textual Case Indexing. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 380–391. Springer, Heidelberg (2005)
3. Jarmasz, M., Szpakowicz, S.: Roget's thesaurus and semantic similarity. In: Proceedings of the International Conference on Recent Advances in NLP (RANLP-03), pp. 212–219 (2003)
4. Lund, K., Burgess, C.: Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research, Methods, Instruments and Computers 28(2), 203–208 (1996)
5. Lemaire, B., Denhière, G.: Effects of High-Order Co-occurrences on Word Semantic Similarity. Current Psychology Letters 18(1) (2006)
6. Kontostathis, A., Pottenger, W.M.: A framework for understanding LSI performance. Information Processing and Management 42(1), 56–73 (2006)
7. Mill, W., Kontostathis, A.: Analysis of the values in the LSI term-term matrix, Technical report, Ursinus College (2004)
8. Mitchell, T.: Machine Learning. Mc Graw Hill International (1997)
9. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
10. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Proc. of ECML, pp. 137–142. ACM Press, New York (1998)
11. Chakraborti, S., Mukras, R., Lothian, R., Wiratunga, N., Watt, S., Harper, D.: Supervised Latent Semantic Indexing using Adaptive Sprinkling. In: Proc. of IJCAI, pp. 1582–1587 (2007)

12. Lamontagne, L.: Textual CBR Authoring using Case Cohesion, in TCBR'06 - Reasoning with Text. In: Proceedings of the ECCBR'06 Workshops, pp. 33–43 (2006)
13. Edmonds, P.: Choosing the word most typical in context using a lexical co-occurrence network. Meeting of the Association for Computational Linguistics, 507–509 (1997)
14. Lenz, M.: Knowledge Sources for Textual CBR Applications. In: Lenz, M. (ed.) Textual CBR: Papers from the 1998 Workshop Technical Report WS-98-12, pp. 24–29. AAAI Press, Stanford (1998)
15. Semeraro, G., Lops, P., Degemmis, M.: WordNet-based User Profiles for Neighborhood Formation in Hybrid Recommender Systems. In: Procs. of Fifth HIS Conference, pp. 291–296 (2005)
16. Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., Chen, Z.: Scalable Collaborative Filtering Using Cluster-based Smoothing. In: Procs. of the 28th ACM SIGIR Conference, pp. 114–121 (2005)
17. Terra, E., Clarke, C.L.A.: Frequency Estimates for Word Similarity Measures. In: Proceedings of HLT-NAACL 2003, Main Papers, pp. 165–172 (2003)
18. Lenz, M., Burkhard, H.-D.: CBR for Document Retrieval - The FAllQ Project. In: Leake, D.B., Plaza, E. (eds.) Case-Based Reasoning Research and Development. LNCS, vol. 1266, pp. 84–93. Springer, Heidelberg (1997)
19. Budanitsky, A.: Lexical semantic relatedness and its application in natural language processing, Technical Report CSRG390, University of Toronto (1999)
20. Patterson, D., Rooney, N., Dobrynin, V., Galushka, M.: Sophia: A novel approach for textual case-based reasoning. In: Proc. of IJCAI, pp. 1146–1153 (2005)
21. Chakraborti, S., Lothian, R., Wiratunga, N., Watt, S.: Exploiting Higher Order Word Associations in Textual CBR, Technical Report, The Robert Gordon University (2007)
22. Wiratunga, N., Massie, S., Lothian, R.: Unsupervised Textual Feature Selection. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 340–354. Springer, Heidelberg (2006)
23. Mori, J., Ishizuka, M., Matsuo, Y.: Extracting Keyphrases To Represent Relations in Social Networks from Web. In: Proc. of the Twentieth IJCAI Conference, pp. 2820–2825 (2007)
24. Boucher-Ryan, P., Bridge, D.: Collaborative Recommending using Formal Concept Analysis. Knowledge-Based Systems 19(5), 309–315 (2006)