Using lexical and relational similarity to classify semantic relations

Diarmuid Ó Séaghdha
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
do242@cl.cam.ac.uk

Ann Copestake
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
aac10@cl.cam.ac.uk

Abstract

Many methods are available for computing semantic similarity between individual words, but certain NLP tasks require the comparison of word pairs. This paper presents a kernel-based framework for application to relational reasoning tasks of this kind. The model presented here combines information about two distinct types of word pair similarity: lexical similarity and relational similarity. We present an efficient and flexible technique for implementing relational similarity and show the effectiveness of combining lexical and relational models by demonstrating state-of-the-art results on a compound noun interpretation task.

1 Introduction

The problem of modelling semantic similarity between words has long attracted the interest of researchers in Natural Language Processing and has been shown to be important for numerous applications. For some tasks, however, it is more appropriate to consider the problem of modelling similarity between pairs of words. This is the case when dealing with tasks involving *relational* or *analogical reasoning*. In such tasks, the challenge is to compare pairs of words on the basis of the semantic relation(s) holding between the members of each pair. For example, the noun pairs (*steel,knife*) and (*paper,cup*) are similar because in both cases the relation N_2 *is made of* N_1 frequently holds between their members. Analogical tasks are distinct from (but not unrelated to) other kinds of "relation extraction" tasks where each data item is tied to a specific sentence context (e.g., Girju et al. (2007)).

One such relational reasoning task is the problem of compound noun interpretation, which has received a great deal of attention in recent years (Girju et al., 2005; Turney, 2006; Butnariu and Veale, 2008). In English (and other languages), the process of producing new lexical items through compounding is very frequent and very productive. Furthermore, the noun-noun relation expressed by a given compound is not explicit in its surface form: a *steel knife* may be a *knife made from steel* but a *kitchen knife* is most likely to be a *knife used in a kitchen*, not a *knife made from a kitchen*. The assumption made by similarity-based interpretation methods is that the likely meaning of a novel compound can be predicted by comparing it to previously seen compounds whose meanings are known. This is a natural framework for computational techniques; there is also empirical evidence for similarity-based interpretation in human compound processing (Ryder, 1994; Devereux and Costello, 2007).

This paper presents an approach to relational reasoning based on combining information about two kinds of similarity between word pairs: *lexical similarity* and *relational similarity*. The assumptions underlying these two models of similarity are sketched in Section 2. In Section 3 we describe how these models can be implemented for statistical machine learning with kernel methods. We present a new flexible and efficient kernel-based framework for classification with relational similarity. In Sections 4 and 5 we apply our methods to a compound interpretation task and demonstrate that combining models of lexical and relational similarity can give state-of-the-art results on a compound noun interpretation task, surpassing the performance attained by either model taken alone. We then discuss previous research on relational similarity, and show that some previously proposed models can be implemented in our framework as special cases. Given the good performance achieved for compound interpretation, it seems likely that the methods presented in this paper can also be applied successfully to other relational reasoning tasks; we suggest some directions for future research in Section 7.

2 Two models of word pair similarity

While there is a long tradition of NLP research on methods for calculating semantic similarity between words, calculating similarity between pairs (or n-tuples) of words is a less well-understood problem. In fact, the problem has rarely been stated explicitly, though it is implicitly addressed by most work on compound noun interpretation and semantic relation extraction. This section describes two complementary approaches for using distributional information extracted from corpora to calculate noun pair similarity.

The first model of pair similarity is based on standard methods for computing semantic similarity between individual words. According to this *lexical similarity* model, word pairs (w_1, w_2) and (w_3, w_4) are judged similar if w_1 is similar to w_3 and w_2 is similar to w_4. Given a measure $wsim$ of word-word similarity, a measure of pair similarity $psim$ can be derived as a linear combination of pairwise lexical similarities:

$$\begin{aligned} &psim((w_1, w_2), (w_3, w_4)) = \\ &\quad \alpha[wsim(w_1, w_3)] \ + \beta[wsim(w_2, w_4)] \end{aligned} \tag{1}$$

A great number of methods for lexical semantic similarity have been proposed in the NLP literature. The most common paradigm for corpus-based methods, and the one adopted here, is based on the *distributional hypothesis*: that two words are semantically similar if they have similar patterns of co-occurrence with other words in some set of contexts. Curran (2004) gives a comprehensive overview of distributional methods.

The second model of pair similarity rests on the assumption that when the members of a word pair are mentioned in the same context, that context is likely to yield information about the relations holding between the words' referents. For example, the members of the pair $(bear, forest)$ may tend to co-occur in contexts containing patterns such as w_1 *lives in the* w_2 and *in the* w_2,*... a* w_1, suggesting that a *LOCATED_IN* or *LIVES_IN* relation frequently holds between bears and forests. If the contexts in which *fish* and *reef* co-occur are similar to those found for *bear* and *forest*, this is evidence that the same semantic relation tends to hold between the members of each pair. A *relational distributional hypothesis* therefore states that two word pairs are semantically similar if their members appear together in similar contexts.

The distinction between lexical and relational similarity for word pair comparison is recognised by Turney (2006) (he calls the former *attributional similarity*), though the methods he presents focus on relational similarity. Ó Séaghdha and Copestake's (2007) classification of information sources for noun compound interpretation also includes a description of lexical and relational similarity. Approaches to compound noun interpretation have tended to use either lexical or relational similarity, though rarely both (see Section 6 below).

3 Kernel methods for pair similarity

3.1 Kernel methods

The kernel framework for machine learning is a natural choice for similarity-based classification (Shawe-Taylor and Cristianini, 2004). The central concept in this framework is the *kernel function*, which can be viewed as a measure of similarity between data items. Valid kernels must satisfy the mathematical condition of *positive semi-definiteness*; this is equivalent to requiring that the kernel function equate to an inner product in some vector space. The kernel can be expressed in terms of a mapping function ϕ from the input space $\mathcal{X}$ to a feature space $\mathcal{F}$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}} \tag{2}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ is the inner product associated with $\mathcal{F}$. $\mathcal{X}$ and $\mathcal{F}$ need not have the same dimensionality or be of the same type. $\mathcal{F}$ is by definition an inner product space, but the elements of $\mathcal{X}$ need not even be vectorial, so long as a suitable mapping function ϕ can be found. Furthermore, it is often possible to calculate kernel values without explicitly representing the elements of $\mathcal{F}$; this allows the use of implicit feature spaces with a very high or even infinite dimensionality.

Kernel functions have received significant attention in recent years, most notably due to the successful application of Support Vector Machines (Cortes and Vapnik, 1995) to many problems. The SVM algorithm learns a decision boundary between two data classes that maximises the minimum distance or *margin* from the training points in each class to the boundary. The geometry of the space in which this boundary is set depends on the kernel function used to compare data items. By tailoring the choice of kernel to the task at hand, the user can use prior knowledge and intuition to improve classification performance.

One useful property of kernels is that any sum or linear combination of kernel functions is itself a valid kernel. Theoretical analyses (Cristianini et al., 2001; Joachims et al., 2001) and empirical investigations (e.g., Gliozzo et al. (2005)) have shown that combining kernels in this way can have a beneficial effect when the component kernels capture different "views" of the data while individually attaining similar levels of discriminative performance. In the experiments described below, we make use of this insight to integrate lexical and relational information for semantic classification of compound nouns.

3.2 Lexical kernels

Ó Séaghdha and Copestake (2008) demonstrate how standard techniques for distributional similarity can be implemented in a kernel framework. In particular, kernels for comparing probability distributions can be derived from standard probabilistic distance measures through simple transformations. These distributional kernels are suited to a data representation where each word w is identified with the a vector of conditional probabilities $(P(c_1|w), \ldots, P(c_{|C|}|w))$ that defines a distribution over other terms c co-occurring with w. For example, the following positive semi-definite kernel between words can be derived from the well-known Jensen-Shannon divergence:

$$\begin{aligned} & k_{jsd}(w_1, w_2) = \\ & \quad -\sum_c [P(c|w_1)\log_2(\frac{P(c|w_1)}{P(c|w_1)+P(c|w_2)}) \\ & \quad + P(c|w_2)\log_2(\frac{P(c|w_2)}{P(c|w_1)+P(c|w_2)})] \quad (3) \end{aligned}$$

A straightforward method of extending this model to word pairs is to represent each pair (w_1, w_2) as the concatenation of the co-occurrence probability vectors for w_1 and w_2. Taking k_{jsd} as a measure of word similarity and introducing parameters α and β to scale the contributions of w_1 and w_2 respectively, we retrieve the lexical model of pair similarity defined above in (1). Without prior knowledge of the relative importance of each pair constituent, it is natural to set both scaling parameters to 0.5, and this is done in the experiments below.

3.3 String embedding functions

The necessary starting point for our implementation of relational similarity is a means of comparing contexts. Contexts can be represented in a variety of ways, from unordered bags of words to rich syntactic structures. The context representation adopted here is based on *strings*, which preserve useful information about the order of words in the context yet can be processed and compared quite efficiently. *String kernels* are a family of kernels that compare strings $\mathbf{s}$, $\mathbf{t}$ by mapping them into feature vectors $\phi_{String}(\mathbf{s})$, $\phi_{String}(\mathbf{t})$ whose non-zero elements index the subsequences contained in each string.

A *string* is defined as a finite sequence $\mathbf{s} = (s_1, \ldots, s_l)$ of symbols belonging to an alphabet Σ. Σ^l is the set of all strings of length l, and Σ^* is set of all strings or the *language*. A subsequence $\mathbf{u}$ of $\mathbf{s}$ is defined by a sequence of indices $\mathbf{i} = (i_1, \ldots, i_{|\mathbf{u}|})$ such that $1 \leq i_1 < \cdots < i_{|\mathbf{u}|} \leq |\mathbf{s}|$, where $|\mathbf{s}|$ is the length of $\mathbf{s}$. $len(\mathbf{i}) = i_{|\mathbf{u}|} - i_1 + 1$ is the length of the subsequence in $\mathbf{s}$. An *embedding* $\phi_{String} : \Sigma^* \rightarrow \mathbb{R}^{|\Sigma|^l}$ is a function that maps a string s onto a vector of positive "counts" that correspond to subsequences contained in $\mathbf{s}$.

One example of an embedding function is a *gap-weighted embedding*, defined as

$$\phi_{gap_l}(\mathbf{s}) = [\sum_{\mathbf{i}:\mathbf{s}[\mathbf{i}]=\mathbf{u}} \lambda^{len(\mathbf{i})}]_{\mathbf{u}\in\Sigma^l} \quad (4)$$

λ is a decay parameter between 0 and 1; the smaller its value, the more the influence of a discontinuous subsequence is reduced. When $l = 1$ this corresponds to a "bag-of-words" embedding. Gap-weighted string kernels implicitly compute the similarity between two strings $\mathbf{s}$, $\mathbf{t}$ as an inner product $\langle\phi(\mathbf{s}), \phi(\mathbf{t})\rangle$. Lodhi et al. (2002) present an efficient dynamic programming algorithm that evaluates this kernel in $O(l|s||t|)$ time without explicitly representing the feature vectors $\phi(\mathbf{s})$, $\phi(\mathbf{t})$.

An alternative embedding is that used by Turney (2008) in his PairClass system (see Section 6). For the PairClass embedding ϕ_{PC}, an n-word context

$$[0\!-\!1 \text{ words}]\ N_{1|2}\ [0\!-\!3 \text{ words}]\ N_{1|2}\ [0\!-\!1 \text{ words}]$$

containing target words N_1, N_2 is mapped onto the 2^{n-2} patterns produced by substituting zero or more of the context words with a wildcard $*$. Unlike the patterns used by the gap-weighted embedding these are not truly discontinuous, as each wildcard must match exactly one word.

3.4 Kernels on sets

String kernels afford a way of comparing individual contexts. In order to compute the relational similarity of two pairs, however, we do not want to associate each pair with a single context but rather with the set of contexts in which they appear together. In this section, we use string embeddings to define kernels on sets of strings.

One natural way of defining a kernel over sets is to take the average of the pairwise basic kernel values between members of the two sets A and B. Let k_0 be a kernel on a set $\mathcal{X}$, and let $A, B \subseteq \mathcal{X}$ be sets of cardinality $|A|$ and $|B|$ respectively. The *averaged kernel* is defined as

$$k_{ave}(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} k_0(a, b) \qquad (5)$$

This kernel was introduced by Gärtner et al. (2002) in the context of multiple instance learning. It was first used for computing relational similarity by Ó Séaghdha and Copestake (2007). The efficiency of the kernel computation is dominated by the $|A| \times |B|$ basic kernel calculations. When each basic kernel calculation $k_0(a, b)$ has significant complexity, as is the case with string kernels, calculating k_{ave} can be slow.

A second perspective views each set as corresponding to a probability distribution, and takes the members of that set as observed samples from that distribution. In this way a kernel on distributions can be cast as a kernel on sets. In the case of sets whose members are strings, a string embedding ϕ_{String} can be used to estimate a probability distribution over subsequences for each set by taking the normalised sum of the feature mappings of its members:

$$\phi_{Set}(A) = \frac{1}{Z} \sum_{\mathbf{s} \in A} \phi_{String}(\mathbf{s}) \qquad (6)$$

where Z is a normalisation factor. Different choices of ϕ_{String} yield different relational similarity models. In this paper we primarily use the gap-weighted embedding ϕ_{gap_l}; we also discuss the PairClass embedding ϕ_{PC} for comparison.

Once the embedding ϕ_{Set} has been calculated, any suitable inner product can be applied to the resulting vectors, e.g. the linear kernel (dot product) or the Jensen-Shannon kernel defined in (3). In the latter case, which we term k_{jsd} below, the natural choice for normalisation is the sum of the entries in $\sum_{\mathbf{s} \in A} \phi_{String}(\mathbf{s})$, ensuring that $\phi_{Set}(A)$ has unit L_1 norm and defines a probability distribution. Furthermore, scaling $\phi_{Set}(A)$ by $\frac{1}{|A|}$, applying L_2 vector normalisation and applying the linear kernel retrieves the averaged set kernel $k_{ave}(A, B)$ as a special case of the distributional framework for sets of strings.

Instead of requiring $|A||B|$ basic kernel evaluations for each pair of sets, distributional set kernels only require the embedding $\phi_{Set}(A)$ to be computed once for each set and then a single vector inner product for each pair of sets. This is generally far more efficient than the kernel averaging method. The significant drawback is that representing the feature vector for each set demands a large amount of memory; for the gap-weighted embedding with subsequence length l, each vector potentially contains up to $|A|\binom{|\mathbf{s}_{max}|}{l}$ entries, where $\mathbf{s}_{max}$ is the longest string in A. In practice, however, the vector length will be lower due to subsequences occurring more than once and many strings being shorter than $\mathbf{s}_{max}$.

One way to reduce the memory load is to reduce the lengths of the strings used, either by retaining just the part of each string expected to be informative or by discarding all strings longer than an acceptable maximum. The PairClass embedding function implicitly restricts the contexts considered by only applying to strings where no more than three words occur between the targets, and by ignoring all non-intervening words except single ones adjacent to the targets. A further technique is to trade off time efficiency for space efficiency by computing the set kernel matrix in a blockwise fashion. To do this, the input data is divided into blocks of roughly equal size – the size that is relevant here is the sum of the cardinalities of the sets in a given block. Larger block sizes b therefore allow faster computation, but they require more memory. In the experiments described below, b was set to 5,000 for embeddings of length $l = 1$ and $l = 2$, and to 3,000 for $l = 3$.

4 Experimental setup for compound noun interpretation

4.1 Dataset

The dataset used in our experiments is Ó Séaghdha and Copestake's (2007) set of 1,443 compound nouns extracted from the British National Corpus (BNC).[1] Each compound is annotated with one of

[1]The data are available from `http://www.cl.cam.ac.uk/~do242/resources.html`.

six semantic relations: *BE*, *HAVE*, *IN*, *AGENT*, *INSTRUMENT* and *ABOUT*. For example, *air disaster* is labelled *IN* (*a disaster in the air*) and *freight train* is labelled *INSTRUMENT* (*a train that carries freight*). The best previous classification result on this dataset was reported by Ó Séaghdha and Copestake (2008), who achieved 61.0% accuracy and 58.8% F-score with a purely lexical model of compound similarity.

4.2 General Methodology

All experiments were run using the LIBSVM Support Vector Machine library.[2] The one-versus-all method was used to decompose the multiclass task into six binary classification tasks. Performance was evaluated using five-fold cross-validation. For each fold the SVM cost parameter was optimised in the range $(2^{-6}, 2^{-4}, \ldots, 2^{12})$ through cross-validation on the training set.

All kernel matrices were precomputed on near-identical machines with 2.4 Ghz 64-bit processors and 8Gb of memory. The kernel matrix computation is trivial to parallelise, as each cell is independent. Spreading the computational load across multiple processors is a simple way to reduce the real time cost of the procedure.

4.3 Lexical features

Our implementation of the lexical similarity model uses the same feature set as Ó Séaghdha and Copestake (2008). Two corpora were used to extract co-occurrence information: the written component of the BNC (Burnard, 1995) and the Google Web 1T 5-Gram Corpus (Brants and Franz, 2006). For each noun appearing as a compound constituent in the dataset, we estimate a co-occurrence distribution based on the nouns in coordinative constructions. Conjunctions are identified in the BNC by first parsing the corpus with RASP (Briscoe et al., 2006) and extracting instances of the `conj` grammatical relation. As the 5-Gram corpus does not contain full sentences it cannot be parsed, so regular expressions were used to extract coordinations. In each corpus, the set of co-occurring terms is restricted to the 10,000 most frequent conjuncts in that corpus so that each constituent distribution is represented with a 10,000-dimensional vector. The probability vector for the compound is created by appending the two constituent vectors, each scaled by 0.5 to weight both constituents equally and ensure that the new vector sums to 1. To perform classification with these features we use the Jensen-Shannon kernel (3).[3]

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

4.4 Relational features

To extract data for computing relational similarity, we searched a large corpus for sentences in which both constituents of a compound co-occur. The corpora used here are the written BNC, containing 90 million words of British English balanced across genre and text type, and the English Gigaword Corpus, 2nd Edition (Graff et al., 2005), containing 2.3 billion words of newswire text. Extraction from the Gigaword Corpus was performed at the paragraph level as the corpus is not annotated for sentence boundaries, and a dictionary of plural forms and American English variants was used to expand the coverage of the corpus trawl.

The extracted contexts were split into sentences, tagged and lemmatised with RASP. Duplicate sentences were discarded, as were sentences in which the compound head and modifier were more than 10 words apart. Punctuation and tokens containing non-alphanumeric characters were removed. The compound modifier and head were replaced with placeholder tokens `M:n` and `H:n` in each sentence to ensure that the classifier would learn from relational information only and not from lexical information about the constituents. Finally, all tokens more than five words to the left of the leftmost constituent or more than five words to the right of the rightmost constituent were discarded; this has the effect of speeding up the kernel computations and should also focus the classifier on the most informative parts of the context sentences. Examples of the context strings extracted for the modifier-head pair (*history*,*book*) are *the:a 1957:m pulitizer:n prize-winning:j H:n describe:v event:n in:i american:j M:n when:c elect:v official:n take:v principle:v* and *he:p read:v constantly:r usually:r H:n about:i american:j M:n or:c biography:n*.

This extraction procedure resulted in a corpus of 1,472,798 strings. There was significant variation in the number of context strings extracted for each compound: 288 compounds were associated with 1,000 or more sentences, while 191 were as-

[3] Ó Séaghdha and Copestake (2008) achieve their single best result with a different kernel (the *Jensen-Shannon RBF kernel*), but the kernel used here (the *Jensen-Shannon linear kernel*) generally achieves equivalent performance and presents one fewer parameter to optimise.

	k_{jsd}		k_{ave}	
Length	Acc	F	Acc	F
1	47.9	45.8	43.6	40.4
2	51.7	49.5	49.7	48.3
3	50.7	48.4	50.1	48.6
Σ_{12}	51.5	49.6	48.3	46.8
Σ_{23}	**52.1**	**49.9**	50.9	49.5
Σ_{123}	51.3	49.0	50.5	49.1
ϕ_{PC}	44.9	43.3	40.9	40.0

Table 1: Results for combinations of embedding functions and set kernels

sociated with 10 or fewer and no sentences were found for 45 constituent pairs. The largest context sets were predominantly associated with political or economic topics (e.g., *government official*, *oil price*), reflecting the journalistic sources of the Gigaword sentences.

Our implementation of relational similarity applies the two set kernels k_{ave} and k_{jsd} defined in Section 3.4 to these context sets. For each kernel we tested gap-weighted embedding functions with subsequence length values l in the range $1, 2, 3$, as well as summed kernels for all combinations of values in this range. The decay parameter λ for the subsequence feature embedding was set to 0.5 throughout, in line with previous recommendations (e.g., Cancedda et al. (2003)). To investigate the effects of varying set sizes, we ran experiments with context sets of maximal cardinality $q \in \{50, 250, 1000\}$. These sets were randomly sampled for each compound; for compounds associated with fewer strings than the maximal cardinality, all associated strings were used. For $q = 50$ we average results over five runs in order to reduce sampling variation. We also report some results with the PairClass embedding ϕ_{PC}. The restricted representative power of this embedding brings greater efficiency and we were able to use $q = 5,000$; for all but 22 compounds, this allowed the use of all contexts for which the ϕ_{PC} embedding was defined.

5 Results

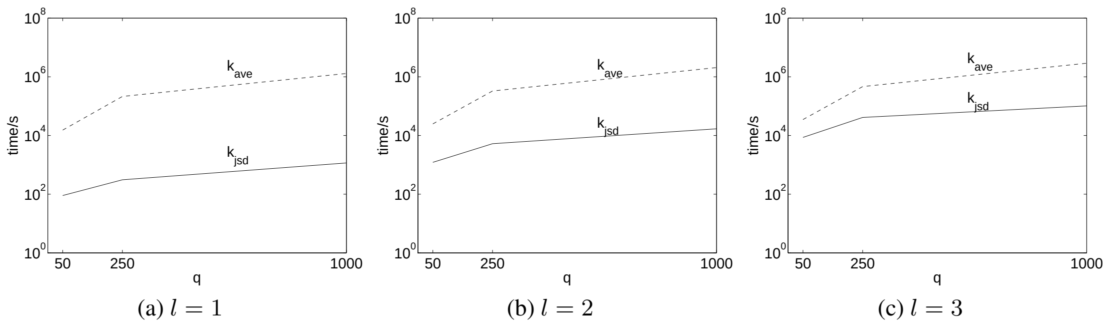
Table 1 presents results for classification with relational set kernels, using $q = 1,000$ for the gap-weighted embedding. In general, there is little difference between the performance of k_{jsd} and k_{ave} with ϕ_{gap_l}; the only statistically significant differences (at $p < 0.05$, using paired t-tests) are between the kernels $k_{l=1}$ with subsequence length $l = 1$ and the summed kernels $k_{\Sigma_{12}} = k_{l=1} + k_{l=2}$. The best performance of 52.1% accuracy, 49.9% F-score is obtained with the Jensen-Shannon kernel k_{jsd} computed on the summed feature embeddings of length 2 and 3. This is significantly lower than the performance achieved by Ó Séaghdha and Copestake (2008) with their lexical similarity model, but it is well above the majority class baseline (21.3%). Results for the PairClass embedding are much lower than for the gap-weighted embedding; the superiority of ϕ_{gap_l} is statistically significant in all cases except $l = 1$.

Results for combinations of lexical co-occurrence kernels and (gap-weighted) relational set kernels are given in Table 2. With the exception of some combinations of the length-1 set kernel, these results are clearly better than the best results obtained with either the lexical or the relational model taken alone. The best result is obtained by the combining the lexical kernel computed on BNC conjunction features with the summed Jensen-Shannon set kernel $k_{\Sigma_{23}}$; this combination achieves 63.1% accuracy and 61.6% F-score, a statistically significant improvement (at the $p < 0.01$ level) over the lexical kernel alone and the best result yet reported for this dataset. Also, the benefit of combining set kernels of different subsequence lengths l is evident; of the 12 combinations presented Table 2 that include summed set kernels, nine lead to statistically significant improvements over the corresponding lexical kernels taken alone (the remaining three are also close to significance).

Our experiments also show that the distributional implementation of set kernels (6) is much more efficient than the averaging implementation (5). The time behaviour of the two methods with increasing set cardinality q and subsequence length l is illustrated in Figure 1. At the largest tested values of q and l (1,000 and 3, respectively), the averaging method takes over 33 days of CPU time, while the distributional method takes just over one day. In theory, k_{ave} scales quadratically as q increases; this was not observed because for many constituent pairs there are not enough context strings available to keep adding as q grows large, but the dependence is certainly superlinear. The time taken by k_{jsd} is theoretically linear in q, but again scales less dramatically in practice. On the other hand k_{ave} is linear in l, while k_{jsd} scales exponentially. This exponential dependence may

	k_{jsd}				k_{ave}			
	BNC		5-Gram		BNC		5-Gram	
Length	Acc	F	Acc	F	Acc	F	Acc	F
1	60.6	58.6	60.3	58.1	59.5	57.6	59.1	56.5
2	61.9*	60.4*	62.6	60.8	62.0	60.5*	61.3	59.1
3	62.5*	60.8*	61.7	59.9	62.8*	61.2**	62.3**	60.8**
Σ_{12}	62.6*	61.0**	62.3*	60.6*	62.0*	60.3*	61.5	59.2
Σ_{23}	**63.1****	**61.6****	62.3*	60.5*	62.2*	60.7*	62.0	60.3
Σ_{123}	62.9**	61.3**	62.6	60.8*	61.9*	60.4*	62.4*	60.6*
No Set	59.9	57.8	60.2	58.1	59.9	57.8	60.2	58.1

Table 2: Results for set kernel and lexical kernel combination. */** indicate significant improvement at the 0.05/0.01 level over the corresponding lexical kernel alone, estimated by paired t-tests.



(a) $l = 1$ (b) $l = 2$ (c) $l = 3$

Figure 1: Timing results (in seconds, log-scaled) for averaged and Jensen-Shannon set kernels

seem worrying, but in practice only short subsequence lengths are used with string kernels. In situations where set sizes are small but long subsequence features are desired, the averaging approach may be more appropriate. However, it seems likely that many applications will be similar to the task considered here, where short subsequences are sufficient and it is desirable to use as much data as possible to represent each set. We note that calculating the PairClass embedding, which counts far fewer patterns, took just 1h21m. For optimal efficiency, it seems best to use a gap-weighted embedding with small set cardinality; averaged across five runs k_{jsd} with $q = 50$ and $l = \Sigma_{123}$ took 26m to calculate and still achieved 47.6% Accuracy, 45.1% F-score.

6 Related work

Turney et al. (2003) suggest combining various information sources for solving SAT analogy problems. However, previous work on compound interpretation has generally used either lexical similarity or relational similarity but not both in combination. Previously proposed lexical models include the WordNet-based methods of Kim and Baldwin (2005) and Girju et al. (2005), and the distributional model of Ó Séaghdha and Copestake (2008). The idea of using relational similarity to understand compounds goes back at least as far as Lebowitz' (1988) RESEARCHER system, which processed patent abstracts in an incremental fashion and associated an unseen compound with the relation expressed in a context where the constituents previously occurred.

Turney (2006) describes a method (*Latent Relational Analysis*) that extracts subsequence patterns for noun pairs from a large corpus, using query expansion to increase the recall of the search and feature selection and dimensionality reduction to reduce the complexity of the feature space. LRA performs well on analogical tasks including compound interpretation, but has very substantial resource requirements. Turney (2008) has recently proposed a simpler SVM-based algorithm for analogical classification called *PairClass*. While it does not adopt a set-based or distributional model of relational similarity, we have noted above that PairClass implicitly uses a feature representation similar to the one presented above as (6) by extracting subsequence patterns from observed co-occurrences of word pair members. Indeed, PairClass can be viewed as a special case of our framework; the differences from the model we have used consist in the use of a different embedding function ϕ_{PC} and a more restricted notion of context, a frequency cutoff to eliminate less common subsequences and the Gaussian kernel to compare vectors. While we cannot compare methods directly as we do not possess the large corpus of 5×10^{10} words used by Turney, we have tested the impact of each of these modifications on our model.[4] None improve performance with our set kernels, but the only statistically significant effect is that of changing the embedding model as reported in section Section 5. Implementing the full PairClass algorithm on our corpus yields 46.2% accuracy, 44.9% F-score, which is again significantly worse than all results for the gap-weighted model with $l > 1$.

In NLP, there has not been widespread use of set representations for data items, and hence set classification techniques have received little attention. Notable exceptions include Rosario and Hearst (2005) and Bunescu and Mooney (2007), who tackle relation classification and extraction tasks by considering the set of contexts in which the members of a candidate relation argument pair co-occur. While this gives a set representation for each pair, both sets of authors apply classification methods at the level of individual set members rather than directly comparing sets. There is also a close connection between the multinomial probability model we have proposed and the pervasive *bag of words* (or *bag of n-grams*) representation. Distributional kernels based on a gap-weighted feature embedding extend these models by using bags of discontinuous n-grams and down-weighting gappy subsequences.

A number of set kernels other than those discussed here have been proposed in the machine learning literature, though none of these proposals have explicitly addressed the problem of comparing sets of strings or other structured objects, and many are suitable only for comparing sets of small cardinality. Kondor and Jebara (2003) take a distributional approach similar to ours, fitting multivariate normal distributions to the feature space mappings of sets A and B and comparing the mappings with the Bhattacharrya vector inner product. The model described above in (6) implicitly fits multinomial distributions in the feature space $\mathcal{F}$; this seems more intuitive for string kernel embeddings that map strings onto vectors of positive-valued "counts". Experiments with Kondor and Jebara's Bhattacharrya kernel indicate that it can in fact come close to the performances reported in Section 5 but has significantly greater computational requirements due to the need to perform costly matrix manipulations.

7 Conclusion and future directions

In this paper we have presented a combined model of lexical and relational similarity for relational reasoning tasks. We have developed an efficient and flexible kernel-based framework for comparing sets of contexts using the feature embedding associated with a string kernel.[5] By choosing a particular embedding function and a particular inner product on subsequence vectors, the previously proposed set-averaging and PairClass algorithms for relational similarity can be retrieved as special cases. Applying our methods to the task of compound noun interpretation, we have shown that combining lexical and relational similarity is a very effective approach that surpasses either similarity model taken individually.

Turney (2008) argues that many NLP tasks can be formulated in terms of analogical reasoning, and he applies his PairClass algorithm to a number of problems including SAT verbal analogy tests, synonym/antonym classification and distinction between semantically similar and semantically associated words. Our future research plans include investigating the application of our combined similarity model to analogical tasks other than compound noun interpretation. A second promising direction is to investigate relational models for unsupervised semantic analysis of noun compounds. The range of semantic relations that can be expressed by compounds is the subject of some controversy (Ryder, 1994), and unsupervised learning methods offer a data-driven means of discovering relational classes.

Acknowledgements

We are grateful to Peter Turney, Andreas Vlachos and the anonymous EACL reviewers for their helpful comments. This work was supported in part by EPSRC grant EP/C010035/1.

[4]Turney (p.c.) reports that the full PairClass model achieves 50.0% accuracy, 49.3% F-score.

[5]The treatment presented here has used a string representation of context, but the method could be extended to other structural representations for which substructure embeddings exist, such as syntactic trees (Collins and Duffy, 2001).

References

Thorsten Brants and Alex Franz, 2006. *Web 1T 5-gram Corpus Version 1.1*. Linguistic Data Consortium.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-06 Interactive Presentation Sessions*.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the Web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

Lou Burnard, 1995. *Users' Guide for the British National Corpus*. British National Corpus Consortium.

Cristina Butnariu and Tony Veale. 2008. A concept-centered approach to noun-compound interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of the 15th Conference on Neural Information Processing Systems (NIPS-01)*.

Corinna Cortes and Vladimir Vapnik. 1995. Support vector networks. *Machine Learning*, 20(3):273–297.

Nello Cristianini, Jaz Kandola, Andre Elisseeff, and John Shawe-Taylor. 2001. On kernel target alignment. Technical Report NC-TR-01-087, NeuroCOLT.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Barry Devereux and Fintan Costello. 2007. Learning to interpret novel noun-noun compounds: Evidence from a category learning experiment. In *Proceedings of the ACL-07 Workshop on Cognitive Aspects of Computational Language Acquisition*.

Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. 2002. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning (ICML-02)*.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479–496.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-07)*.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda, 2005. *English Gigaword Corpus, 2nd Edition*. Linguistic Data Consortium.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of noun compounds using WordNet similarity. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*.

Risi Kondor and Tony Jebara. 2003. A kernel between sets of vectors. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*.

Michael Lebowitz. 1988. The use of memory in text processing. *Communications of the ACM*, 31(12):1483–1502.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Diarmuid Ó Séaghdha and Ann Copestake. 2007. Co-occurrence contexts for noun compound interpretation. In *Proceedings of the ACL-07 Workshop on A Broader Perspective on Multiword Expressions*.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*.

Barbara Rosario and Marti A. Hearst. 2005. Multi-way relation classification: Application to protein-protein interactions. In *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP-05)*.

Mary Ellen Ryder. 1994. *Ordered Chaos: The Interpretation of English Noun-Noun Compounds*. University of California Press, Berkeley, CA.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge.

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the 2003 International Conference on Recent Advances in Natural Language Processing (RANLP-03)*.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*.