Natural Language Engineering **1** (1): 1–27. Printed in the United Kingdom
© 1998 Cambridge University Press

Interpreting compound nouns with kernel methods

DIARMUID Ó SÉAGHDHA and ANN COPESTAKE
Computer Laboratory, University of Cambridge, UK

(*Received 20 March 1995; revised 30 September 1998*)

Abstract

This paper presents a classification-based approach to noun-noun compound interpretation within the statistical learning framework of kernel methods. In this framework, the primary modelling task is to define measures of similarity between data items, formalised as kernel functions. We consider the different sources of information that are useful for understanding compounds and proceed to define kernels that compute similarity between compounds in terms of these sources. In particular, these kernels implement intuitive notions of lexical and relational similarity and can be computed using distributional information extracted from text corpora. We report performance on classification experiments with three semantic relation inventories at different levels of granularity, demonstrating in each case that combining lexical and relational information sources is beneficial and gives better performance than either source taken alone. The data used in our experiments are taken from general English text, but our methods are also applicable to other domains and potentially to other languages where noun-noun compounding is frequent and productive.

1 Introduction

Compounding is a general procedure used in a great variety of languages to create new words out of old (Bauer 2001). In English it is very common to combine two or more nouns to produce a *noun-noun compound* that functions as a single lexical item. Many such compounds are familiar to all speakers and are assumed to be lexicalised to some degree (*e.g. taxi driver*, *trade union*, *coffee table*), yet completely novel compounds are frequently created and can often be interpreted by language users without conscious deliberation or even awareness of the compound's ambiguity (what is a *pineapple radio*?). Empirical studies have suggested that 3-4% of all words in English text are constituents of a noun-noun compound, while the vast majority of compounds occur very rarely in accordance with a Zipfian distribution of types (Baldwin and Tanaka 2004; Ó Séaghdha 2008).

The frequency of compounds in English means that a natural language processing (NLP) system for high-recall semantic interpretation cannot avoid dealing with them. At the same time, the productivity of compounding means that one cannot hope to store meanings for all possible compounds in a dictionary or lexical database. NLP research on compound interpretation has a long history, going back at least as far as Su (1969) and Russell (1972), and remains an active area of research. One reason for its longevity as an open problem is that the semantic relation holding between a compound's constituents is not directly indicated by its surface form. It is not possible to predict a compound's meaning from its modifier and head words:[1] a *cheese knife* is a *knife for cutting cheese* but a *kitchen knife* is not a *knife for cutting kitchens* and a *cheese sandwich* is not a *sandwich for cutting cheese*. Rather, the relational meaning must be "decoded" by recruiting a combination of lexical clues, analogical reasoning, knowledge about language use and knowledge about the world. This makes compound understanding an archetypal challenge for computational semantics: a wide-ranging semantic competence is necessary in order for an automatic system to perform well.

Early research assumed the existence of a detailed, manually constructed semantic lexicon, based on which logical interpretation rules could be defined. In common with most other NLP problems, the past two decades have seen a shift towards statistical methods for semantic analysis in order to attain broad coverage and robust performance in the face of prevalent lexical and relational ambiguity. In this paper we cast compound interpretation as a classification problem: the task is to select the correct relation label from a fixed inventory. This is in line with most recent NLP research on compounds (Lauer 1995; Nastase and Szpakowicz 2003; Girju et al.2005; Turney 2006). However, some researchers have proposed an alternative *paraphrase-based* paradigm, in which compound interpretation is seen as the problem of finding one or more paraphrases that make explicit the relation expressed by a compound (Nakov 2008; Butnariu et al.2010). We believe that the two approaches are complementary, in the sense that labels from a fixed relation inventory generally correspond to coarse-grained groupings of paraphrases; for example, the coarse label *HAVE* might subsume the paraphrases *X belongs to Y*, *Y owns X*, *Y has X*, *X is part of Y*, *etc.*

The statistical learning approach we adopt is based on *kernel methods*, whereby the fundamental modelling construct is a notion of similarity between items formalised as a function with particular mathematical properties (the *kernel function*). Applying this framework to the task at hand amounts to an analogical hypothesis of compound interpretation: we assume the relational meaning of a given compound can be predicted, at least in part, by knowledge about the meanings of similar compounds. Some cognitive linguists and psycholinguists have argued that human compound interpretation is an analogical or similarity-driven process (Ryder 1994; Gagné and Shoben 2002; Devereux and Costello 2007), suggesting that an analogical conception of the problem is justified. The kernel framework is well-suited to classification with heterogeneous information sources; this property is fundamental to our approach as we show that combining *lexical* and *relational* information about compounds can

[1] Following standard assumptions about the syntax of English, we describe the first word in a two-noun compound as the "modifier" and the second word as the "head". We also assume that for non-lexicalised compositional compounds, the semantic type of the compound and that of the head agree.

yield levels of performance that significantly exceed the performance attained by either kind of information on its own.

The kernel methods applied here are based on those previously proposed by Ó Séaghdha and Copestake (2008; 2009).[2] As well as an integrated description of these methods, this paper describes experiments with a wider collection of feature sets and an expanded experimental design involving classification at multiple levels of semantic granularity. In doing so, we report the best performance yet achieved on the compound dataset of Ó Séaghdha (2008), surpassing the results previously achieved by other researchers and by ourselves.

2 Information sources for compound interpretation

Understanding a noun-noun compound comes easily to human language users but it involves applying a rich resource of heterogeneous information: the discourse context, "world knowledge" about entities and how they interact, lexical knowledge about familiar compounds. In order to exploit a similar range of knowledge sources for automatic interpretation we might consider the following kinds of information and related measures of similarity:

Lexical information: Information about the individual constituent words of a compound.

Relational information: Information about how the entities denoted by a compound's constituents typically interact in the world.

Contextual information: Information derived from the context in which a compound occurs.

Table 1 gives examples of each kind of information as extracted for two compounds used in our experiments.

Adopting a similarity-based perspective on interpretation, the use of lexical information assumes that compounds whose constituents are similar will tend to have similar relational meanings. If we know that a *history book* is a *book about history*, we might assume that a *geography article* is an *article about geography* given the lexical similarity between *history* and *geography* and between *book* and *article*. The task of computing lexical similarity is a much-studied one in NLP; among the most successful data-driven techniques are *distributional methods*, which relate the meaning of a word to its patterns of co-occurrence in text corpora. The fundamental idea here is to associate each word with a numerical vector that measures the strength of association between that word and each member of a set of *co-occurrence contexts*, which may correspond to words or to basic syntactic structures. When the co-occurrence vector of a word is a normalised vector of counts that sums to 1, this is equivalent to estimating a probability distribution over co-occurrence contexts for that word. The literature on distributional semantics is voluminous and cannot be adequately summarised here, but many comprehensive overviews exist (Curran

[2] Performance figures in this paper are not identical to comparable figures in our previous papers due to minor differences in corpus processing.

Table 1: Information sources for sample compounds; lexical information is represented by syntactic coordination features, relational information by sets of context strings and contextual information by the BNC sentence from which the compound was originally extracted

kidney disease	
LEXICAL:	
mod	*liver*:460 *heart*:225 *lung*:186 *brain*:148 *spleen*:100
head	*cancer*:964 *disorder*:707 *syndrome*:483 *condition*:440 *injury*:427
RELATIONAL:	*Stagnant water breeds fatal diseases of liver and kidney such as hepatitis*
	Chronic disease causes kidney function to worsen over time until dialysis is needed
	This disease attacks the kidneys, liver, and cardiovascular system
CONTEXT:	*These include the elderly, people with chronic respiratory disease, chronic heart disease, kidney disease and diabetes, and health service staff*
holiday village	
LEXICAL:	
mod	*weekend*:507 *sunday*:198 *holiday*:180 *day*:159 *event*:115
head	*municipality*:9417 *parish*:4786 *town*:4526 *hamlet*:1634 *city*:1263
RELATIONAL:	*He is spending the holiday at his grandmother's house in the mountain of Busang in the Vosges region*
	The Prime Minister and his family will spend their holidays in Vernet, a village of 2,000 inhabitants located about 20 kilometers south of Toulouse
	Other holiday activities include a guided tour of Panama City, a visit to an Indian village and a helicopter tour
CONTEXT:	*For FFr100m ($17.5m), American Express has bought a 2% stake in Club Méditerranée, a French group that ranks third among European tour operators, and runs holiday villages in exotic places*

2003; Padó and Lapata 2007; Turney and Pantel 2010). In this paper we implement lexical similarity by estimating co-occurrence distributions for each constituent of a compound and comparing the corresponding constituents of two compounds using a kernel defined on probability distributions (Section 3.2).

Whereas measures of distributional lexical similarity consider the co-occurrences of each constituent of a word pair separately, relational similarity is based on the contexts in which both constituents appear together. Relational similarity has also been called *analogical similarity*, *e.g.* by Turney (Turney 2006; Turney 2008). The underlying intuition is that when nouns N_1 and N_2 are mentioned in the same context, that context is likely to yield information about the relations that hold between those nouns' referents in the world. For example, contexts mentioning *brick* and *wall* might frequently describe a *wall made of bricks* or a *wall constructed with bricks*, while contexts mentioning *wood* and *floor* might describe a *floor made of wood* or a *wall constructed with wood.* A *relational distributional hypothesis* would therefore state that two word pairs are semantically similar if their members appear together in similar contexts. In order to compute relational similarity between compounds in our kernel-based framework, we extract sets of contexts for each modifier-head pair in our compound dataset and use a kernel on sets of strings to compare compounds (Section 3.3).

As a general principle in NLP, the context in which a pair of word tokens occur is a valuable source of information about the semantic relation that holds between them. As one might expect, context features such as surrounding words play an important role in tasks such as semantic role labelling or semantic relation classification in the vein of ACE or SemEval-07 Task 4 (ACE 2008; Girju et al.2007). Compound interpretation can indeed be seen as a special case of semantic relation identification; however, context features are of little value for understanding compounds, as demonstrated empirically by Ó Séaghdha and Copestake (2007). This should not be surprising, as it is a defining characteristic of compounds that information about the semantic relation between constituents is not made explicit in context but rather left for the hearer to "unpack". Treated as a single lexical item, the distributional profile of a compound is primarily defined by its head noun rather than by its relational semantics. As a result we do not use contextual similarity in the experiments described here, even though the compound dataset we use has been annotated in context.

Most statistical learning approaches to compound interpretation use lexical information, relational information or both. Lexical information is usually extracted from a lexical resource such as WordNet or Roget's Thesaurus (Nastase and Szpakowicz 2003; Girju et al.2005; Kim and Baldwin 2005; Tratz and Hovy 2010); apart from our own work, the only use of distributional features that we are aware of is by Nastase et al. (Nastase et al.2006). Our eschewal of WordNet and other manually constructed resources is motivated in part by a scientific interest in the potential for learning semantics using purely distributional information and partly by a concern for portability to domains and languages where broad-coverage lexical resources do not exist. Relational information extracted from corpora (including the World Wide Web) has been applied to the interpretation problem by Lauer (1995), Turney (2006; 2008), Nakov and Hearst (2008) and Tratz and Hovy (2010). One further source of similarity information that we do not consider (but could easily add) is morphological information, including suffix and prefix features (Tratz and Hovy 2010).

3 Kernel methods for classification

3.1 Overview

Having sketched our concepts of lexical and relational similarity, we proceed in this section to provide technical details of the methods we use to implement them. *Kernel methods* is the name given to a statistical modelling paradigm in which data items are viewed by the learning algorithm in terms of their similarity to other items (Shawe-Taylor and Cristianini 2004). This similarity is captured through a function known as the *kernel function* (or simply *kernel*). Kernel-based learning has a number of advantages, including the facilitation of efficient methods for non-linear classification and classification with complex feature spaces, and has become very popular in NLP and many other disciplines. Furthermore, kernel algorithms are modular in the sense that switching to a new kernel does not require a new learning algorithm. While traditional feature-based learning can also exploit rich feature mappings, one cannot simply "slot in" a new inner product geometry as we do here without altering the learning algorithm.

A *kernel* is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$ which is equivalent to an inner product $\langle \cdot, \cdot \rangle_\mathcal{F}$ in some inner product space $\mathcal{F}$ (the *feature space*):

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_\mathcal{F} \tag{1}$$

where ϕ is a mapping from $\mathcal{X}$ to $\mathcal{F}$. For example, the polynomial kernel of degree $l = 3$, $k_{P3}(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + R)^3$ defined on input vectors of dimension $d = 2$ corresponds to an inner product in the space whose $\binom{d+l}{l} = 10$ dimensions are the monomials of degree i, $1 \leq i \leq 3$, hence $\phi(\mathbf{x}) = \phi(x_1, x_2) = [x_1^3, x_2^3, x_1^2 x_2, x_1 x_2^2, x_1^2, x_2^2, x_1 x_2, x_1, x_2]$. Note that we do not need to represent the 10-dimensional vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ as the kernel can be computed directly in 2-d space. Given a set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathcal{X}\}$ and a kernel k, the $n \times n$ matrix K with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is called the *kernel matrix* or *Gram matrix*. It follows from Mercer's Theorem (Mercer 1909) that a valid kernel on a set $\mathcal{X}$ is defined by any symmetric finitely positive semi-definite function, *i.e.* a function for which the Gram matrix of function values calculated on any finite set $X \subseteq \mathcal{X}$ satisfies

$$\mathbf{v}' K \mathbf{v} \geq 0, \forall \mathbf{v} \in \mathbb{R}^n \tag{2}$$

The space of functions satisfying the positive semi-definite (psd) property (2) is large. The Euclidean dot product $\langle \cdot, \cdot \rangle_{\mathbb{R}^d}$, also known as the *linear kernel* in the context of classification, is a kernel for which the input and spaces are the same and ϕ is the identity mapping. Other kernels, such as the *Gaussian kernel* $k = \exp(-\beta \sum_d (x_{id} - x_{jd})^2)$, correspond to an inner product in an infinite-dimensional space. Furthermore, the elements of the input set $\mathcal{X}$ need not be vectors. In Sections 3.2 and 3.3 below, we consider classes of kernels on probability distributions, strings and sets of strings that will prove useful for compound classification. For other aspects of kernel methods, the book by Shawe-Taylor and Cristianini (2004) is recommended as a comprehensive guide.

Given training data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ drawn identically and independently from some set $\mathcal{X}$, with corresponding labels $Y = \{y_1, \ldots, y_n\}$ belonging to a set $\mathcal{Y}$, the *supervised classification* task is to learn a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that best predicts the unknown label y_j for a data point $\mathbf{x}_j \in \mathcal{X}$ that was not seen in training. In order to render learning feasible, f is usually restricted to a particular functional class; for binary classification (*i.e.* $\mathcal{Y} = \{-1, 1\}$) we might define $f(\mathbf{x}) = \operatorname{sign} g(\mathbf{x})$ where $g : \mathcal{X} \mapsto \mathbb{R}$ belongs to the class of linear functions:

$$\begin{aligned} g(\mathbf{x}) &= \langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathbb{R}^d} + b & (3) \\ &= \sum_i w_i \phi(\mathbf{x})_i + b & (4) \end{aligned}$$

Here we have introduced a function $\phi : \mathcal{X} \mapsto \mathbb{R}^d$ that maps data points to a d-dimensional *feature vector* of real numbers and a vector $\mathbf{w}$ of *feature weights* that capture the predictive power attributed to each feature. $\langle \cdot, \cdot \rangle_{\mathbb{R}^d}$ is the inner product in $\mathbb{R}^d$, *i.e.* the dot product. The vector $\mathbf{w}$ defines a hyperplane $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathbb{R}^d} = 0$ in $\mathbb{R}^d$; the sign of $g(\mathbf{x})$ indicates the side of this hyperplane on which the point $\mathbf{x}$ lies, while the absolute value of $g(\mathbf{x})$ indicates the distance from $\mathbf{x}$ to the hyperplane.

A dataset is *linearly separable* if a hyperplane exists such that all instances with label -1 are on one side and all instances with label 1 are on the other. Linear separability is usually an unrealistic condition for complex problems. The *support vector machine* (SVM) is a classifier that aims to find the "best" decision boundary in that violations of the boundary are minimised and the separation between classes (the "margin") is maximised (Cortes and Vapnik 1995). The (dual) SVM optimisation problem is:

$$\boldsymbol{\alpha}^* = \operatorname{argmax}_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = -\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i) \phi(\mathbf{x}_j) \tag{5}$$

$$\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0, \; \sum_{i=1}^{n} \alpha_i = 1,$$
$$0 \leq \alpha_i \leq C, \text{ for all } i = 1, \ldots, l$$

where the parameter C controls the tradeoff between maximising the margin and tolerating misclassifications. The dual objective function $W(\boldsymbol{\alpha})$ in (5) depends on the training examples $\mathbf{x}_i$ only through their inner products $\phi(\mathbf{x}_i)\phi(\mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{F}}$. This suggests that we can use support vector machines to do classification in a kernel feature space by rewriting the objective as

$$W(\boldsymbol{\alpha}) = -\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{6}$$

where k is a valid kernel on $\mathcal{X}$. The SVM classifier optimising this objective will learn a decision function $f(\mathbf{x}) = \operatorname{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathcal{F}} + b)$ that is linear in the codomain $\mathcal{F}$ of ϕ but nonlinear in the input space $\mathcal{X}$.

Support vector machines have been shown to achieve state-of-the-art performance on a great number of classification problems in NLP and other fields. Good performance with SVMs is contingent on the choice of an appropriate kernel function. The kernel chosen determines both the feature mapping ϕ and the inner product used to compare data points in feature space. If a kernel function induces a mapping into a space where the data classes are well separated, then learning a decision boundary in that space will be easy. Conversely, if the feature space mapping of the data does not contain discriminative information, SVM classification will perform poorly. Hence if we can use a kernel function tailored to the prior knowledge we have about our classification problem, we expect to do better than we would with a less appropriate kernel. In the following sections we discuss how our intuitions about lexical and relational similarity can be applied to improve performance on the compound classification task.

3.2 Kernels on probability distributions

The most commonly used kernels on vectors of real numbers (including co-occurrence vectors) are the linear and Gaussian kernels. It can be shown that these kernels are related to the Euclidean L_2 distance by two transformations that relate psd kernels k to squared metrics $\tilde{k}$, also known as negative semi-definite kernels (Berg et al.1984):

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tilde{k}(\mathbf{x}_i, \mathbf{x}_0) + \tilde{k}(\mathbf{x}_j, \mathbf{x}_0) - \tilde{k}(\mathbf{x}_i, \mathbf{x}_j) - \tilde{k}(\mathbf{x}_0, \mathbf{x}_0),\ \mathbf{x}_0 \in \mathcal{X} \text{ (linear)} \tag{7a}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\beta \cdot \tilde{k}(\mathbf{x}_i, \mathbf{x}_j)),\ \beta > 0 \text{ (RBF)} \tag{7b}$$

Taking $\tilde{k}$ to be the square of the L_2 distance, *i.e.* $\tilde{k}_{L_2} = \sum_d (x_{id} - x_{jd})^2$ and setting $\mathbf{x}_0$ to be the zero vector, the linear transformation (7a) yields a function that is twice the linear kernel and the RBF transformation (7b) yields the Gaussian kernel:

$$k_{L_2(linear)}(\mathbf{x_i}, \mathbf{x_j}) = \sum_i x_{ii} x_{ji} \tag{8}$$

$$k_{L_2(RBF)}(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\beta \cdot \tilde{k}_{L_2}(\mathbf{p}, \mathbf{q})\right) \tag{9}$$

Research on distributional lexical semantics has shown that the L_2 distance performs relatively poorly compared to other distance and similarity measures (Lee 1999). Hein and Bousquet (2005) describe a family of kernels that relate to well-known distance measures on probability distributions, including the Jensen-Shannon Divergence (JSD), L_1 distance and Hellinger Divergence, using the transformations (7a) and (7b). Ó Séaghdha and Copestake (2008) observe that these kernels include a number of well-established distributional similarity measures and demonstrate empirically that performance on classification tasks involving lexical similarity can be improved by using such *distributional kernels*. A representative example is the Jensen-Shannon Divergence

$$\tilde{k}_{JSD}(\mathbf{p}, \mathbf{q}) = \sum_i p_i \log_2 \left(\frac{2p_i}{p_i + q_i}\right) + q_i \log_2 \left(\frac{2q_i}{p_i + q_i}\right) \tag{10}$$

from which linear and RBF kernels can be derived using (7a) and (7b), respectively:

$$k_{JSD(linear)}(\mathbf{p},\mathbf{q}) = -\sum_i p_i \log_2\left(\frac{p_i}{p_i+q_i}\right) + q_i \log_2\left(\frac{q_i}{p_i+q_i}\right) \tag{11a}$$

$$k_{JSD(RBF)}(\mathbf{p},\mathbf{q}) = \exp\left(-\beta \cdot \tilde{k}_{JSD}(\mathbf{p},\mathbf{q})\right) \tag{11b}$$

The function (11a) had previously been suggested as a distributional similarity measure by Lin (1999) (though not in the context of classification). Note that not all popular distributional similarity measures can be used as kernels, as some do not fulfill the psd condition (2); for example, an asymmetric similarity measure such as Kullback-Leibler Divergence cannot be psd.

To compare two words w, w' with a distributional kernel, we compare their estimated co-occurrence distributions. As our goal is to apply distributional kernels to compound interpretation, we must proceed from comparing individual words to comparing pairs of words. A lexical kernel on compounds can be defined as the normalised sum of a kernel function on heads and a kernel function on modifiers:

$$k_{lex}((w_m, w_h), (w'_m, w'_h)) = \frac{1}{2}k(w_m, w'_m) + \frac{1}{2}k(w_h, w'_h) \tag{12}$$

In practice, (12) is equivalent to a kernel between vectors produced by concatenating a normalised head vector and a normalised modifier vector for each compound and computing a distributional kernel on pairs of concatenated vectors.

The development of new kernels on distributions is an area of active research in the machine learning community. In addition to the family of kernels described by Hein and Bousquet, further distributional kernels have been proposed by Lafferty and Lebanon (2005), Martins et al. (2009) and Agarwal and Daumé III (2011). The experiments described in Ó Séaghdha (2008) and preliminary experiments on some of these more recent proposals suggest that while most of these distributional kernels consistently outperform Euclidean linear and RBF kernels, their performance on semantic classification tasks is often similar. As the focus of this paper is on integrating semantic information sources rather than a comprehensive investigation of alternative kernels on distributions, we use only the L_2 and JSD linear kernels (8) and (11a) in our experiments below.

3.3 Kernels on strings and sets

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. Once an embedding is defined, any inner product can be used to compare items.

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 $\mathbf{i} = (i_1, \dots, i_{|\mathbf{u}|})$ 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* $\phi_{String} : \Sigma^* \to \mathbb{R}^{|\Sigma|^l}$ is a function that maps a string s onto a vector of positive "counts" that correspond to subsequences of exactly l symbols contained in $\mathbf{s}$.

One example of an embedding function is a *gap-weighted embedding*, which incorporates information about non-contiguous as well as contiguous subsequences. This is an intuitively useful property for semantic classification, as words in a sentence need not be adjacent to be semantically related. The gap-weighted embedding is defined as

$$\phi_{gap_l}(\mathbf{s}) = [\sum_{\mathbf{i}:\mathbf{s}[\mathbf{i}]=\mathbf{u}} \lambda^{len(\mathbf{i})}]_{\mathbf{u}\in\Sigma^l} \tag{13}$$

λ is a decay parameter between 0 and 1; the smaller its value, the more the influence of a discontinuous subsequence is reduced. The length parameter l controls the length of the subsequences that defined the embedding. When $l = 1$, (13) corresponds to a "bag-of-words" embedding; when $l = 2$, the embedding extracts all discontinuous bigrams. 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 the need to represent the feature vectors $\phi(\mathbf{s})$, $\phi(\mathbf{t})$; their kernel implicitly uses the L_2 linear kernel for comparing strings, but a distributional linear kernel such as (11a) can also be used at the cost of some efficiency as Lodhi et al.'s algorithm cannot generally be used.

An alternative embedding is that used by Turney (2008) in his PairClass system for analogical classification. 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. This restriction does mean that PairClass embeddings are more efficient to compute than gap-weighted embeddings, but as we show in Section 3.3 below comes at a significant cost in terms of discriminative capacity.

Relational similarity relies on comparing sets of contexts, *i.e.* *sets* of strings. In order to derive useful kernels on such sets, we use an embedding function for sets that uses an embedding of set members as a building block. A natural approach is to map a set to the normalised aggregate of the embeddings of its members:

$$\phi_{Set}(A) = \frac{1}{Z}\sum_{\mathbf{s}\in A} \phi_{String}(\mathbf{s}) \tag{14}$$

If the normalising term Z is such that the embeddings $\phi_{Set}(A)$ have unit Euclidean norm and we compare embeddings with the Euclidean dot product, then we retrieve the averaged kernel of Gärtner et al. (2002) with the string kernel of Lodhi et al. (2002) as base kernel. Alternatively we can normalise the embedding so that it sums to 1, defining a multinomial probability distribution over subsequences. Under this perspective, context sets are viewed as samples from underlying probability distributions and it is natural to use distributional kernels to compare sets. In this paper we implement relational similarity as the JSD linear kernel computed between embeddings of context sets produced with set embedding (14) and string embedding ϕ_{gap_l} (13) or the PairClass embedding ϕ_{PC}:

$$k_{rel}((w_m, w_h), (w'_m, w'_h)) = k_{JSD(linear)}\left(\phi_{Set}(S(w_m, w_h)), \phi_{Set}(S(w'_m, w'_h))\right) \quad (15)$$

where $S(w_m, w_h)$ is the set of context strings associated with the word pair (w_m, w_h).

3.4 Kernel combination

The set of kernels is closed under addition and multiplication by a positive scalar (Shawe-Taylor and Cristianini 2004). It follows that a linear combination of l kernels with positive weights $\boldsymbol{\mu}$ is also a valid kernel:

$$k_{comb}(\mathbf{x}_i, \mathbf{x}_j) = \sum_l \mu_l k_l(\mathbf{x}_i, \mathbf{x}_j), \boldsymbol{\mu} \in \mathbb{R}^l_+ \quad (16)$$

This method of kernel combination gives us a way to integrate heterogeneous sources of information inside a kernel-based classifier. As regards the choice of combination weights $\boldsymbol{\mu}$, Joachims et al. (2001) state that these parameters are relatively unimportant so long as the elementary kernels being combined each have comparable performance. Good results can often be achieved with uniform weights.

Multiple kernel learning (MKL) is the name usually given to the task of optimising the combination weights in (16). Finding the best $\boldsymbol{\mu}$ is a non-trivial optimisation problem for more than a small number of kernels, as it is not practical to perform an exhaustive search. As far as we are aware, MKL methods have not previously been applied in NLP research; we performed experiments using a multiclass adaptation of the two-stage algorithm proposed by Cortes et al. (2010), but our findings were inconclusive. MKL can give a small boost in performance but it is not always reliable, especially for larger label sets, and our best results were achieved through uniform weighting.

4 Experimental design

4.1 Compound dataset

Our experiments use the dataset of 1,443 unique two-noun English compounds described at length in (Ó Séaghdha 2008). These compounds were randomly sampled from a collection of 430,555 types (covering over 1.5 million tokens) harvested from the written component of the British National Corpus (Burnard 1995) and annotated in context for their semantic relations. The inventory of semantic relations consists of six coarse-grained relations: *BE*, *HAVE*, *IN*, *ACTOR*, *INST(RUMENT)* and *ABOUT*. Each is a binary relation; a compound expressing a particular relation is assumed to fill one argument slot with its head and the other with its modifier. With the exception of *BE*, the argument slots for each relation are semantically distinct and it is useful to note which constituent fills which slot. Thus *pig pen* would be labelled $IN_{(M,H)}$ as the head denotes the location and the modifier denotes the located entity; *pen pig* would be labelled $IN_{(H,M)}$ as the head and modifier roles are reversed.

Furthermore, each compound in the dataset is annotated with the annotation rule that licenses the assignment of its coarse-grained label. For example, the rules licensing IN distinguish between spatially located objects (*pig pen*), spatially located events (*air disaster*), temporally located objects (*evening edition*) and temporally located events (*dawn attack*). The table in Appendix A summarises the various annotation rules and their distribution over the dataset. The design of the inventory was motivated by a desire to avoid inconsistent or overlapping relations and by a concern for general linguistic and psycholinguistic principles. The dataset is freely available online, as are the detailed guidelines used by the annotators.[3]

Ó Séaghdha (2008) reports an inter-annotator agreement analysis performed on a more complex task which contained as a part the semantic annotation used here but also involved the detection of extraction errors and non-compositional or lexicalised compounds. Overall agreement was 66.2% (Cohen's $\kappa = 0.62$) for coarse-grained annotation, 64.0% ($\kappa = 0.61$) for coarse-grained labels and role order) and 58.8% ($\kappa = 0.56$). Further analysis by Ó Séaghdha suggests that agreement on the semantic relation annotation is probably a few points higher (estimated $\kappa = 0.68$) as a significant amount of disagreement relates to decisions about lexicalisation. These agreement figures are not as high as those reported for some other tasks but are comparable to those reported in the literature on semantic annotation of compounds (Tratz and Hovy 2010).

Prior work on this dataset has restricted itself to the coarse-grained annotation with six semantic relation labels (Ó Séaghdha and Copestake 2008; Ó Séaghdha and Copestake 2009; Tratz and Hovy 2010). In this paper we consider the tasks of classification at three levels of granularity:

Coarse: predict coarse-grained relations (6 labels).

Directed: predict coarse-grained relations and which constituents fill which relation slots (11 labels).

Fine: predict fine-grained labels corresponding to annotation rules and which constituents fill which relation slots (37 labels).

Not all potential labels are well-represented in the dataset; for example, there are only five examples of $ABOUT_{(H,M)}$ (*e.g. sitcom family*). In our classification experiments all labels occurring 10 times or fewer in the dataset were excluded, with the result that in practice the **Directed** and **Fine** inventories contain 10 and 27 labels respectively.

[3] The data can be downloaded from `http://www.cl.cam.ac.uk/~do242/Resources/1443_Compounds.tar.gz`; the guidelines are available at `http://www.cl.cam.ac.uk/~do242/Publications/guidelines.pdf`

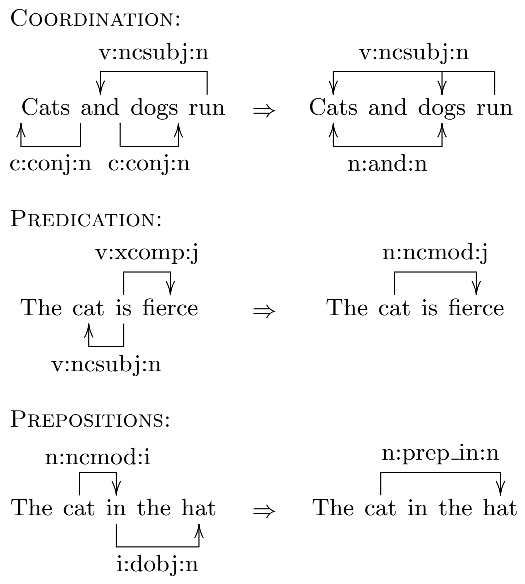


Fig. 1: Dependency graph preprocessing

4.2 Corpora

In order to extract data for the lexical and relational kernels we used three source corpora:

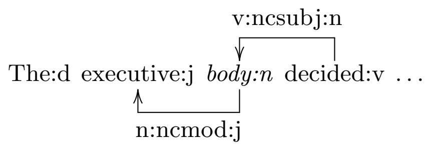
- The written component of the British National Corpus (**BNC**), consisting of 90 million words (Burnard 1995), lemmatised, POS-tagged and parsed with the RASP toolkit (Briscoe et al.2006).
- A Wikipedia dump (**WIKI**), consisting of almost 1 billion words, collected by Clark et al. (2009), lemmatised, POS-tagged and parsed with the fast Combinatory Categorial Grammar (CCG) pipeline described by Clark et al.
- The English Gigaword Corpus, 2nd Edition (**GW**), consisting of 2.3 billion words (Graff et al.2005), lemmatised and tagged with RASP but not parsed.

These corpora are also used to produce two larger corpora by concatenation: **BNC+WIKI** and **BNC+GW**.

The **BNC** and **BNC+WIKI** corpora were used to extract distributional lexical information based on grammatical relations (GRs) in the parser output. Although the **BNC** and **WIKI** corpora were processed with different parsers, both the RASP and CCG parsers can produce grammatical relations in the same format and therefore their outputs are interoperable. In order to maximise the yield of semantically salient information, the GR graph for each sentence was preprocessed with the transformations specified in Figure 1. All parts of speech other than nouns, verbs, adjectives, prepositions and adverbs were ignored, as were stopwords, tokens of fewer than three characters and the semantically weak lemmas *be* and *have*. The following feature sets were extracted for each noun w that appears as a constituent in the compound dataset:

Coordination: All nouns w' appearing linked to w in a coordinative structure (labelled *n:and:n* in Figure 1). This is the same feature specification as in Ó Séaghdha and Copestake (2008); it is very sparse but rich in semantic class information. For example, if a noun is frequently coordinated with *cat*, *dog* and *goldfish*, then it is most likely a household animal.

All GRs: All tuples (w', r) where w' is a word appearing as the dependent of w in the GR graph and r is the GR label on the edge connecting w and w', and all tuples (w', r^{-1}) where w' appears as the head of w. For example, given the GR graph fragment



the set of context features for the noun *body* is $\{(executive, j{:}ncmod^{-1}{:}n), (decide, v{:}ncsubj{:}n)\}$, where $ncmod^{-1}$ denotes that *body* stands in an inverse non-clausal modifier relation to *executive* (we assume that nouns are the heads of their adjectival modifiers).

GR clusters: Rather than using the very sparse **All GRs** feature space, we can use clusters of GRs induced by applying the "topic modelling" approach of Ó Séaghdha and Korhonen (2011). This method, based on the Latent Dirichlet Allocation model proposed for document analysis by Blei et al. (2003), associates each word with a multinomial distribution over latent variables and associates each latent variable with a multinomial distribution over GR tuples. Given that the number of latent variables $|Z|$ is generally much lower than the number of GR tuples, the parameters of a word's distribution over latent variables can be interpreted as defining a low-dimensional distribution over clusters of GR tuples. In order to estimate the parameters of this model, we use the efficient Gibbs sampling algorithm of Yao et al. (2009) and estimate the probability $P(z|w)$ that a GR involving word w is associated with latent variable z as the proportion of GR instances involving w in the training corpus to which z is assigned in the final sampling state. We run the Gibbs sampler for 1,000 iterations and set $|Z| = 1,000$. We have not investigated the effect of modifying $|Z|$, though it would be feasible to combine feature sets with different parameterisations to exploit multiple granularities or to combine feature sets derived from multiple sampling runs to reduce variance.

The BNC+GW corpus was used to extract sets of string contexts for each pair of compound constituents. For each compound in the dataset, the set of sentences in the combined corpus containing both constituents of the compound was identified. As the Gigaword Corpus contains many duplicate and near-duplicate articles, duplicate sentences were discarded so that repeated strings would not dominate the similarity estimates. Sentences in which the head and modifier words were more than 10 words apart were also discarded, as it is plausible that there is no direct relation between the words in such sentences. The 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. Punctuation and tokens containing non-alphanumeric characters were removed. 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 set 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 pulitzer: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

this:d H:n will:v appeal:v to:i lover:n of:i military:j M:n
but:c its:a characterisation:n often:r seem:v

you:p will:v enter:v the:a H:n of:i M:n as:c patriot:n
museveni:n say:v

in:i the:a past:n many:d H:n have:v be:v publish:v on:i the:a
M:n of:i mongolia:n but:c the:a new:j

subject:n no:a surprise:n be:v a:a M:n of:i the:a american:j
comic:j H:n something:p about:i which:d he:p be:v

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 associated 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.

4.3 Classification setup

All experiments were performed using the SVM training algorithm implemented in LIBSVM.[4] We decompose the multiclass classification problem into a set of binary classification problems through the *one-against-all* method: a single classifier is trained for each class label, treating data instances with that class as positive examples and all other instances as negative, and the final prediction for a test instance is the label for which that instance has the greatest positive (or least negative) distance from the corresponding decision boundary. In each cross-validation fold, the SVM cost parameter C is optimised through a grid search over the set $\{2^{-4}, 2^{0}, \ldots, 2^{12}\}$,

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

selecting the value that gives the best 10-fold cross-validation performance on that fold's training set. For each feature set and each set embedding vector, L_1 normalisation (normalisation to unit sum) was performed before computing the JSD kernel; for the experiments with the L_2 linear kernels, L_2 normalisation (normalisation to unit Euclidean norm) was performed.

We compute evaluation results for five-fold cross-validation on the 1,443-compound dataset. As well as prediction accuracy (the proportion of compounds that were correctly labelled) we also report macro-averaged F-Score, the average of the standard F-Score measure calculated individually for each label. Accuracy rewards good performance on the most frequent labels, while macro-averaged F-Score rewards performance on all labels equally. The best previously reported results for this dataset are 63.1% accuracy, 61.6 F-Score (Ó Séaghdha and Copestake 2009) and 63.6% accuracy (Tratz and Hovy 2010). The random-guess baseline is 16.3% accuracy for the Coarse label set, 10.0% accuracy for the Directed label set and 3.7% accuracy for the Fine label set. Where we wish to measure the statistical significance of a difference between two methods we use paired t-tests (two-tailed, df $= 4$) on the cross-validation folds, following the recommendation of Dietterich (1998).

5 Results

5.1 Results for lexical feature kernels

Table 2 presents results for the three lexical feature sets **Coordination**, **All GRs** and **GR Clusters**, using distributional information from the **BNC** and **BNC+WIKI** corpora and the JSD and L_2 linear kernels. Performance is notably lower for the fine-grained label set than on the coarse-grained set; this is not unexpected, as there are four times as many labels and consequentially less positive training data for each label. However, performance on the **Directed** label set is in some cases quite close to that on **Coarse**.

For almost all combinations of feature set and label set, features extracted from the larger **BNC+WIKI** corpus give better performance than **BNC** (the only exceptions occur with the L_2 kernel and fine-grained label set). This is as we would expect, given that a larger corpus contains more information and will be less affected by sparsity of co-occurrences. No one feature set outperforms the others throughout, though the richer **All GRs** and **GR Clusters** sets tend to score better than **Coordination** features alone.

Comparing performance with the JSD and L_2 linear kernels, the distributional kernel is the clear winner, outperforming the standard linear kernel on every combination of feature and label sets. The majority (19/36) of improvements are statistically significant, with many more coming very close to significance. This corroborates the finding of Ó Séaghdha and Copestake (2008) that the JSD kernel is usually a better choice than the standard linear kernel for semantic classification tasks.

Table 2: Comparison of JSD and L_2 linear kernels across feature sets. */** denote significant difference at the 0.05/0.01 level

	Coarse		Directed		Fine	
	Acc	F	Acc	F	Acc	F
BNC/Coordination						
JSD	59.4	57.4	57.9	52.7	45.2	41.6
L_2	58.0	55.7	55.4	49.9	42.2	39.5
Significance			**		*	
BNC/All GRs						
JSD	61.3	59.3	59.0	53.1	46.8	42.4
L_2	56.9	54.4	55.1	49.5	42.6	38.6
Significance	*	*	*		**	**
BNC/GR Clusters						
JSD	62.2	60.0	62.1	57.1	48.0	44.8
L_2	60.4	58.4	58.6	53.7	42.0	38.8
Significance	*		*		**	**
BNC+WIKI/Coord						
JSD	60.9	59.0	**62.2**	**57.4**	48.5	43.9
L_2	59.7	57.5	58.7	53.2	45.9	41.7
Significance						
BNC+WIKI/All GRs						
JSD	62.7	60.6	62.1	56.1	**51.2**	**47.1**
L_2	59.3	57.1	58.3	52.7	45.4	41.6
Significance			**		**	**
BNC+WIKI/GR Clusters						
JSD	**63.0**	**61.0**	61.1	55.2	49.4	45.7
L_2	59.0	56.9	56.3	49.7	44.6	40.8
Significance	**	**	**	**	*	

5.2 Results for relational kernels

Table 3 presents results for classification with relational kernels. Results are reported for settings of the subsequence length parameter l in the range $\{1, 2, 3\}$ as well as additive combinations (denoted Σ) of gap-weighted embeddings corresponding to different subsequence lengths and the PairClass embedding adopted from Turney

Table 3: Results for relational string-set kernels

	Coarse		Directed		Fine	
Embedding	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score
1	48.6	45.7	40.5	35.2	28.6	25.6
2	**52.7**	**50.7**	**50.2**	**44.3**	**37.8**	29.2
3	50.8	48.5	48.1	40.0	34.8	24.2
Σ_{12}	52.3	50.0	49.2	43.3	36.8	**30.5**
Σ_{23}	52.2	49.9	49.8	42.8	37.5	28.5
Σ_{123}	52.0	49.8	49.8	43.1	**37.8**	29.7
PairClass	42.8	41.2	42.1	37.5	30.9	27.0

(2008). In all cases performance is worse than that achieved by the lexical kernels, indicating that relational information (at least as implemented here) is not sufficient for state-of-the-art compound classification. Kernels with $l = 2$ and the combined kernels generally perform best, across all label sets; the difference between these kernels and the PairClass kernel is consistently significant at the $p < 0.01$ level.

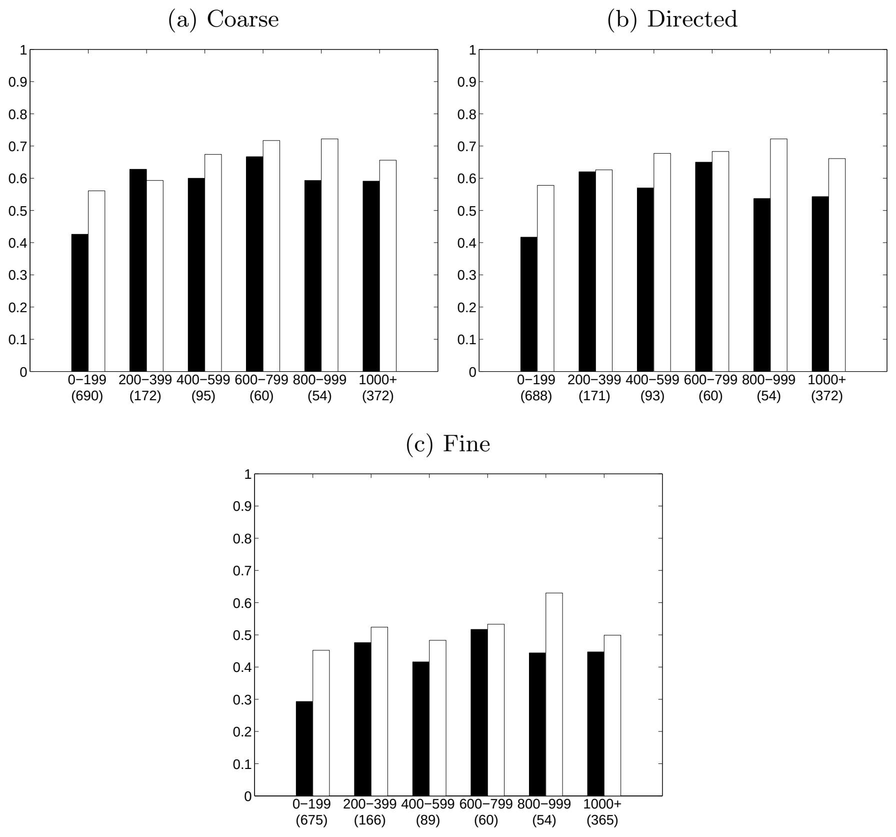
One obvious explanation for the relatively poor performance with relational information alone is data sparsity. As previously noted, 191 compounds are associated with 10 or fewer context strings; in these cases there is very little relational information on which to base a prediction. Figure 2 illustrates the effect of context set size on performance for representative relational and lexical kernels. For the former, we use the $\sum_{123}$ combination, but similar effects are observed for the other subsequence length settings. Across all label sets, classification accuracy with relational kernels drops by about 20% in absolute terms for compounds with fewer than 200 associated context strings. For compounds with more than 200 context strings performance is more or less stable, at a level that is comparable to that achieved by the lexical kernels.

5.3 Results for combined kernels

Table 4 presents results achieved by combining lexical and relational kernels, taking the best-performing lexical feature specifications **All GRs** and **GR clusters**.[5] The benefits of combination are clear: in the vast majority of cases, performance with a combined kernel is better than performance with the relevant lexical kernel on its own. The results are particularly good for combinations involving the summed

[5] We omit results for the **Coordination** feature sets for reasons of space; these follow the same pattern as the results shown.

Fig. 2: The effect of context set size on classification accuracy using the $\sum_{123}$ relational kernel (left, black) and the lexical JSD kernel with **BNC+WIKI/Coord** features (right, white)



relational kernels Σ_{23} and Σ_{123}; across 24 combinations of lexical feature set and label set, only four fail to contribute a statistically significant improvement on at least one evaluation measure. The best results on the **Coarse** label set – 65.4% Accuracy, 64.0 F-Score, achieved by combining the **BNC+WIKI/GR Clusters** lexical kernel with the summed Σ_{123} relational kernel – are higher than all previously reported performances on this dataset.

Table 5 analyses performance for each individual relation in the **Coarse** label set, taking competitive representatives from the lexical, relational and combination kernel categories. The most difficult relations to classify are *BE* and *HAVE*; these are also the least frequent relations in the dataset. *IN* and *ACTOR* seem to be the easiest. The lexical kernel results are better than the relational kernel results for every relation, and the combined kernel results are better than either for every relation except *ACTOR*, where the lexical kernel does very slightly better.

Table 4: Results for combined lexical and relational kernels: */** denote significant improvement over the corresponding lexical-only result at the 0.05/0.01 level

	BNC				BNC+WIKI			
	All GRs		**GR Clusters**		**All GRs**		**GR Clusters**	
	Acc	F	Acc	F	Acc	F	Acc	F
Coarse labels								
Σ_1	60.9	58.6	62.6	60.4	62.8	61.2	63.1	61.5
Σ_2	62.0	60.2	64.0**	62.2**	64.2	62.8	64.0	62.6
Σ_3	62.6	60.7	63.1*	61.4	63.9	62.1	63.8	62.4
Σ_{12}	63.0	61.2	63.4	61.9	64.7*	63.0	64.4*	63.0
Σ_{23}	64.0	62.1	64.5*	62.9*	64.8*	63.2*	64.8*	63.4
Σ_{123}	62.9	61.2	64.4**	62.7*	65.1*	63.4*	**65.4****	**64.0***
Lex only	61.3	59.3	62.2	60.0	62.7	60.6	63.0	61.0
Directed labels								
Σ_1	59.3	54.0	62.4	58.4	62.4	56.7	61.2	56.3
Σ_2	60.8	55.3	63.4	58.7	64.0	58.3*	62.7	57.4
Σ_3	60.6	55.1*	63.5	58.9	63.8*	57.9*	63.3*	58.4
Σ_{12}	60.5	55.1*	64.0*	59.4	63.8*	58.2**	62.9	57.9*
Σ_{23}	61.4	55.3*	65.0**	59.8	64.4	58.4**	64.0*	58.4
Σ_{123}	61.1	55.1*	**65.1***	**60.3**	64.9*	58.9**	64.4	59.1*
Lex only	59.0	53.1	62.1	57.1	62.1	56.1	61.1	55.2
Fine labels								
Σ_1	47.8*	43.3	49.7*	45.5	51.7*	47.5	51.0	46.9
Σ_2	48.7**	43.2	51.4*	46.4	53.3	47.3	52.0*	46.9*
Σ_3	48.3**	42.6	51.5*	46.2	53.2	47.0	52.2**	46.8
Σ_{12}	49.5*	43.9	51.8*	46.8	52.5*	46.5	52.7*	46.8*
Σ_{23}	49.3*	42.7	52.5**	46.4	**53.9***	47.3*	52.7	46.4
Σ_{123}	49.5*	42.9	52.7**	46.9	52.5	46.5	53.5*	**47.6***
Lex only	46.8	42.4	48.0	44.8	51.2	47.1	49.4	45.7

Table 5: Macro-averaged Precision, Recall and F-Score by coarse label for lexical kernel (**BNC+WIKI/GR Clusters**), relational kernel (Σ_{123}) and their combination

	Lexical			Relational			Combined		
	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
BE	55.0	43.5	48.5	38.2	31.4	34.5	**59.5**	**50.8**	**54.8**
HAVE	51.0	39.7	44.6	49.2	30.2	37.4	**55.7**	**46.7**	**50.8**
IN	67.9	71.4	69.6	53.0	68.5	59.8	**70.3**	**72.1**	**71.2**
ACTOR	**68.7**	**77.1**	**72.7**	59.4	66.9	62.9	68.2	76.3	72.0
INST	64.6	65.0	64.8	51.4	49.6	50.5	**64.7**	**67.7**	**66.2**
ABOUT	61.4	**70.8**	65.8	53.5	53.5	53.5	**67.5**	**70.8**	**69.1**
Overall	61.4	61.3	61.0	50.8	50.0	49.8	**64.3**	**64.1**	**64.0**

5.4 Head-only and modifier-only prediction

Table 6 reports results using lexical information about compound heads only or compound modifiers only. These results were attained with the *GR clusters* features and BNC+WIKI corpus, though results with other feature sets and corpora follow the same pattern.

The relative importance of modifier and head words for compound comprehension has been the subject of some debate in psycholinguistics. For example, Gagné and Shoben's (1997) CARIN (*Competition Among Relations In Nominals*) model affords a privileged role to modifiers in determining the range of possible interpretations, and Gagné (2002) finds that meanings can be primed by compounds with semantically similar modifiers but not by similar heads. However, other authors have challenged these findings, including Devereux and Costello (2005), Estes and Jones (2006) and Raffray et al. (2007). While not proposing that human methods of interpretation are directly comparable to machine methods, we suggest that classification performance with head- and modifier-only features can give valuable insight into the discriminative potential of each constituent.

Table 6 shows that overall F-Score and accuracy attained with head features exceed those attained with modifier features by about 10 points for all label granularities. However, head performance is in turn 10-15 points worse than performance with both sets of features, indicating that both do contain valuable information. Of the coarse-grained relations, *BE* is the only relation better predicted with modifier features than with head features. A partial explanation for this last result is that compounds where the modifier is a substance or material frequently express *BE* (annotation rule 2.1.1.2 in Appendix A), for example *corduroy trousers*, *ice crystal*, *copper wire*. On the other hand, modifier information is very weak at recognising

Table 6: Classification results using modifier-only and head-only lexical information (**BNC+WIKI/GR Clusters**)

	Modifier Only				Head Only			
	Acc	Pre	Rec	F	Acc	Pre	Rec	F
BE		**46.7**	**51.8**	**49.1**		38.1	19.4	25.7
HAVE		18.5	11.1	13.8		**48.7**	**38.2**	**42.8**
IN		46.7	**56.8**	51.2		**52.2**	54.2	**53.2**
ACTOR		40.6	42.8	41.6		**53.7**	**69.9**	**60.8**
INST		44.3	45.5	44.9		**51.4**	**54.5**	**52.9**
ABOUT		33.5	29.6	31.4		**57.7**	**66.7**	**61.8**
Overall (coarse)	40.9	38.4	39.6	38.7	**52.1**	**50.3**	**50.4**	**49.5**
Overall (directed)	34.9	29.4	31.1	29.9	**47.9**	**39.8**	**42.6**	**40.2**
Overall (fine)	22.0	19.5	20.0	19.7	**33.4**	**28.3**	**29.4**	**28.5**

instances of *HAVE*, *ACTOR* and *ABOUT*, which seem to be predominantly signalled by the head constituent – for example, compounds headed by *book*, *story* and *film* are very likely to encode a topic relation, while compounds headed by *driver* or *salesman* are likely to encode an agentive relation.

6 Conclusion

In this paper we have described a classification-based approach to compound interpretation that integrates lexical and relational information in a kernel combination framework. We have experimented with three different granularities of semantic relation labels, demonstrating in every case that performance with a combination of information sources is superior to that achieved with individual sources.

Potential future work includes the investigation of new kernels and feature sets for lexical and relational similarity. We also intend to evaluate our methods on other datasets and potentially on other domains and languages; as we do not require specific lexical resources other than a suitable corpus, the methods should be easily portable. Even where high-quality parsing is unavailable or impractical, we hope that reasonable lexical and relational information can still be acquired. More generally, we hope to demonstrate the applicability of our methods to other tasks involving semantic analogies and reasoning about semantic relations.

7 Acknowledgements

We are very grateful to Diane Nicholls for her part in the data annotation, and to Anna Korhonen and Diana McCarthy for feedback on the work. This research was supported by doctoral awards to DÓS by Corpus Christi College Cambridge, the University of Cambridge and the Engineering and Physical Sciences Research Council.

References

ACE, 2008. *Automatic Content Extraction 2008 Evaluation Plan.* Available online at `http://www.nist.gov/speech/tests/ace/ace08/doc/ace08-evalplan.v1.2d.pdf`.

Arvind Agarwal and Hal Daumé III. 2011. Generative kernels for exponential families. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, Ft. Lauderdale, FL.

Timothy Baldwin and Takaaki Tanaka. 2004. Translation by machine of complex nominals: Getting it right. In *Proceedings of the ACL-04 Workshop on Multiword Expressions: Integrating Processing*, Barcelona, Spain.

Laurie Bauer. 2001. Compounding. In Martin Haspelmath, editor, *Language Typology and Language Universals.* Mouton de Gruyter, The Hague.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. 1984. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions.* Springer, Berlin.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

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

Lou Burnard, 1995. *Users' Guide for the British National Corpus.* British National Corpus Consortium, Oxford University Computing Service, Oxford, UK.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2010. Semeval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the SemEval-2 Workshop*, Uppsala, Sweden.

Stephen Clark, Ann Copestake, James R. Curran, Yue Zhang, Aurelie Herbelot, James Haggerty, Byung-Gyu Ahn, Curt Van Wyk, Jessika Roesner, Jonathan Kummerfeld, and Tim Dawborn. 2009. Large-scale syntactic processing: Parsing the web. Technical report, Final Report of the 2009 JHU CLSP Workshop.

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

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. 2010. Two-stage learning kernel algorithms. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel.

James Curran. 2003. *From Distributional to Semantic Similarity.* Ph.D. thesis, School of Informatics, University of Edinburgh.

Barry Devereux and Fintan Costello. 2005. Investigating the relations used in conceptual combination. *Artificial Intelligence Review*, 24(3-4):489–515.

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*, Prague, Czech Republic.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Zachary Estes and Lara L. Jones. 2006. Priming via relational similarity: A *copper horse* is faster when seen through a *glass eye*. *Journal of Memory and Language*, 55(1):89–101.

Christina L. Gagné and Edward J. Shoben. 1997. Influence of thematic relations on the comprehension of modifier-noun combinations. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 23(1):71–87.

Christina L. Gagné and Edward J. Shoben. 2002. Priming relations in ambiguous noun-noun compounds. *Memory and Cognition*, 30(4):637–646.

Christina L. Gagné. 2002. Lexical and relational influences on the processing of novel compounds. *Brain and Language*, 81(1–3):723–735.

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)*, Sydney, Australia.

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)*, Prague, Czech Republic.

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

Matthias Hein and Olivier Bousquet. 2005. Hilbertian metrics and positive definite kernels on probability measures. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS-05)*, Barbados.

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)*, Williamstown, MA.

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)*, Jeju Island, Korea.

John Lafferty and Guy Lebanon. 2005. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Compound Nouns.* Ph.D. thesis, Macquarie University.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, MD.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, MD.

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.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2009. Nonextensive information theoretic kernels on measures. *Journal of Machine Learning Research*, 10:935–975.

J. Mercer. 1909. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London, Series A*, 209:415–446.

Preslav Nakov and Marti A. Hearst. 2008. Solving relational similarity problems using the Web as a corpus. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, Columbus, OH.

Preslav Nakov. 2008. Noun compound interpretation using paraphrasing verbs: Feasibility study. In *Proceedings of the 13th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-08)*, Varna, Bulgaria.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the 5th International Workshop on Computational Semantics (IWCS-03)*, Tilburg, The Netherlands.

Vivi Nastase, Jelber Sayyad Shirabad, Marina Sokolova, and Stan Szpakowicz. 2006. Learning noun-modifier semantic relations with corpus-based and WordNet-based features. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, Boston, MA.

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*, Prague, Czech Republic.

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

Diarmuid Ó Séaghdha and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, Athens, Greece.

Diarmuid Ó Séaghdha and Anna Korhonen. 2011. Probabilistic models of similarity in syntactic context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, Edinburgh, UK.

Diarmuid Ó Séaghdha. 2008. *Learning Compound Noun Semantics*. Ph.D. thesis, University of Cambridge. Published as University of Cambridge Computer Laboratory Technical Report 735.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Claudine N. Raffray, Martin J. Pickering, and Holly P. Branigan. 2007. Priming the interpretation of noun-noun compounds. *Journal of Memory and Language*, 57(3):380–395.

Sylvia Weber Russell. 1972. Semantic categories of nominals for conceptual dependency analysis of natural language. Computer Science Department Report CS-299, Stanford University.

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.

Stanley Y. W. Su. 1969. A semantic theory based upon interactive meaning. Computer Sciences Technical Report #68, University of Wisconsin.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

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)*, Manchester, UK.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-09)*, Paris, France.

A Semantic relations used in the dataset

Rule	Definition	Order	Example	Count	Total
BE	N_1/N_2 *is* N_2/N_1				**191**
2.1.1.1	Identity		*guide dog*		110
2.1.1.2	Substance-Form		*rubber wheel*		50
2.1.1.3	Similarity		*cat burglar*		31
HAVE	N_1/N_2 *has* N_2/N_1				**199**
2.1.2.1	Possessor-Possession	(M,H)	*family firm*	27	31
		(H,M)	*hotel owner*	4	
2.1.2.2	Experiencer-Condition	(M,H)	*member expectation*	10	11
		(H,M)	*coma victim*	1	
2.1.2.3	Object-Property	(M,H)	*grass scent*	54	57
		(H,M)	*colour illustration*	3	
2.1.2.4	Whole-Part	(M,H)	*car tyre*	51	74
		(H,M)	*shelf unit*	23	
2.1.2.5	Group-Member	(M,H)	*union member*	11	26
		(H,M)	*lecture course*	15	
IN	N_1/N_2 *is located in* N_2/N_1				**308**
2.1.3.1	Object-Spatial Location	(M,H)	*fruit dish*	84	188
		(H,M)	*street gang*	104	
2.1.3.2	Event-Spatial Location	(M,H)	*building site*	32	58
		(H,M)	*air disaster*	26	
2.1.3.3	Object-Temporal Location	(M,H)	*policy period*	2	26
		(H,M)	*winter plumage*	24	
2.1.3.4	Event-Temporal Location	(M,H)	*birth date*	17	36
		(H,M)	*weekend trip*	19	

Rule	Definition	Order	Example	Count	Total
ACTOR	*N_1/N_2 is a sentient participant in an event N_2/N_1 or an event involving N_2/N_1*				**236**
2.1.4.1	Participant-Event	(M,H)	*committee discussion*	31	68
		(H,M)	*writing team*	37	
2.1.4.2	Participant1-Participant2	(M,H)	*rate payer*	42	168
		(H,M)	*cowboy hat*	126	
INST	*N_1/N_2 is a non-sentient participant in an event N_2/N_1 or an event involving N_2/N_1*				**266**
2.1.5.1	Participant-Event	(M,H)	*ozone depletion*	100	164
		(H,M)	*operating system*	64	
2.1.5.2	Participant1-Participant2	(M,H)	*computer model*	71	102
		(H,M)	*air filter*	31	
ABOUT	*N_1/N_2 is about N_2/N_1*				**243**
2.1.6.1	Object-Topic	(M,H)	*sitcom family*	3	119
		(H,M)	*history book*	116	
2.1.6.2	Collection-Topic	(M,H)	–	0	7
		(H,M)	*waterway museum*	7	
2.1.6.3	Mental Activity-Focus	(M,H)	*mystery man*	2	93
		(H,M)	*liberation struggle*	91	
2.1.6.4	Charge-Commodity	(M,H)	–	0	24
		(H,M)	*property tax*	24	