Co-occurrence Contexts for Noun Compound Interpretation

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

Contextual information extracted from corpora is frequently used to model semantic similarity. We discuss distinct classes of context types and compare their effectiveness for compound noun interpretation. Contexts corresponding to word-word similarity perform better than contexts corresponding to relation similarity, even when relational co-occurrences are extracted from a much larger corpus. Combining word-similarity and relation-similarity kernels further improves SVM classification performance.

1 Introduction

The compound interpretation task is frequently cast as the problem of classifying an unseen compound noun with one of a closed set of relation categories. These categories may consist of lexical paraphrases, such as the prepositions of Lauer (1995), or deeper semantic relations, such as the relations of Girju et al. (2005) and those used here. The challenge lies in the fact that by their very nature compounds do not give any surface realisation to the relation that holds between their constituents. To identify the difference between *bread knife* and *steel knife* it is not sufficient to assign correct word-senses to *bread*, *steel* and *knife*; it is also necessary to reason about how the entities referred to interact in the world. A common assumption in data-driven approaches to the problem is that compounds with semantically similar constituents will encode similar relations. If a hearer knows that a *fish knife* is a *knife used to eat fish*, he/she might conclude that the novel compound *pigeon fork* is a *fork used to eat pigeon* given that *pigeon* is similar to *fish* and *knife* is similar to *fork*. A second useful intuition is that word pairs which co-occur in similar contexts are likely to enter into similar relations.

In this paper, we apply these insights to identify different kinds of contextual information that capture different kinds of similarity and compare their applicability using medium- to large-sized corpora. In keeping with most other research on the problem,[1] we take a supervised learning approach to compound interpretation.

2 Defining Contexts for Compound Interpretation

When extracting corpus information to interpret a compound such as *bread knife*, there are a number of context types that might plausibly be of interest:

1. The contexts in which instances of the compound type appear (type similarity); e.g., all sentences in the corpus that contain the compound *bread knife*.
2. The contexts in which instances of each constituent appear (word similarity); e.g., all sentences containing the word *bread* or the word *knife*.
3. The contexts in which both constituents appear together (relation similarity); e.g., all sentences containing both *bread* and *knife*.
4. The context in which the particular compound token was found (token similarity).

[1] Such as Girju et al. (2005), Girju (2006), Turney (2006). Lapata and Keller's (2004) unsupervised approach is a notable exception.

Proceedings of the Workshop on A Broader Perspective on Multiword Expressions, pages 57–64,
Prague, June 2007.

A simple but effective method for exploiting these contexts is to count features that co-occur with the target items in those contexts. Co-occurrence may be defined in terms of proximity in the text, lexical patterns, or syntactic patterns in a parse graph. We can parameterise our notion of context further, for example by enforcing a constraint that the co-occurrence correspond to a particular type of grammatical relation or that co-occurrence features belong to a particular word class.[2]

Research in NLP frequently makes use of one or more of these similarity types. For example, Culotta and Sorensen (2004) combine word similarity and relation similarity for relation extraction; Gliozzo et al. (2005) combine word similarity and token similarity for word sense disambiguation. Turney (2006) discusses word similarity (which he calls ”attributional similarity”) and relation similarity, but focusses on the latter and does not perform a comparative study of the kind presented here.

The experiments described here investigate type, word and relation similarity. However, token similarity clearly has a role to play in the interpretation task, as a given compound type can have a different meaning in different contexts – for example, a *school book* can be *a book used in school*, *a book belonging to a school* or *a book about a school*. As our data have been annotated in context, we intend to model this dynamic in future work.

3 Experimental Setup

3.1 Data

We used the dataset of 1443 compounds whose development is described in Ó Séaghdha (2007). These compounds have been annotated in their sentential contexts using the six deep semantic relations listed in Table 1. On the basis of a dual-annotator study, Ó Séaghdha reports agreement of 66.2% ($\hat{\kappa} = 0.62$) on a more general task of annotating a noisy corpus and estimated agreement of 73.6% ($\hat{\kappa} = 0.68$) on annotating the six relations used here. These figures are superior to previously reported results on annotating compounds extracted from corpora. Always choosing the most frequent class (IN) would give accuracy of 21.34%, and we use this as a baseline for our experiments.

Relation	Distribution	Example
BE	191 (13.24%)	*steel knife*, *elm tree*
HAVE	199 (13.79%)	*street name*, *car door*
IN	308 (21.34%)	*forest hut*, *lunch time*
INST	266 (18.43%)	*rice cooker*, *bread knife*
ACTOR	236 (16.35%)	*honey bee*, *bus driver*
ABOUT	243 (16.84%)	*fairy tale*, *history book*

Table 1: The 6 relation classes and their distribution in the dataset

3.2 Corpus

The written section of the British National Corpus,[3] consisting of around 90 million words, was used in all our experiments. This corpus is not large compared to other corpora used in NLP, but it has been manually compiled with a view to a balance of genre and should be more representative of the language in general than corpora containing only newswire text. Furthermore, the compound dataset was also extracted from the BNC and information derived from it will arguably describe the data items more accurately than information from other sources. However, this information may be very sparse given the corpus' size. For comparison we also use a 187 million word subset of the English Gigaword Corpus (Graff, 2003) to derive relational information in Section 6. This subset consists of every paragraph in the Gigaword Corpus belonging to articles tagged as 'story' and containing both constituents of a compound in the dataset, whether or not they are compounded there. Both corpora were lemmatised, tagged and parsed with RASP (Briscoe et al., 2006).

3.3 Learning Algorithm

In all our experiments we use a one-against-all implementation of the Support Vector Machine.[4] Except for the work described in Section 6.2 we used the linear kernel $K(x, y) = x \cdot y$ to compute similarity between vector representations of the data items. The linear kernel consistently achieved superior performance to the more flexible Gaussian kernel in a range tests, presumably due to the sensitivity of

[2]A flexible framework for this kind of context definition is presented by Padó and Lapata (2003).

[3]http://www.natcorp.ox.ac.uk/

[4]The software used was LIBSVM (Chang and Lin, 2001).

the Gaussian kernel to its parameter settings.[5] One-against-all classification (training one classifier per class) performed better than one-against-one (training one classifier for each pair of classes). We estimate test accuracy by 5-fold cross-validation and within each fold we perform further 5-fold cross-validation on the training set to optimise the single SVM parameter *C*. An advantage of the linear kernel is that learning is very efficient. The optimisation, training and testing steps for each fold take from less than a minute on a single processor for the sparsest feature vectors to a few hours for the most dense, and the folds can easily be distributed across machines.

4 Word Similarity

Ó Séaghdha (2007) investigates the effectiveness of word-level co-occurrences for compound interpretation, and the results presented in this section are taken from that paper. Co-occurrences were identified in the BNC for each compound constituent in the dataset, using the following context definitions:

win5, win10: Each word within a window of 5 or 10 words on either side of the item is a feature.

Rbasic, Rmod, Rverb, Rconj: These feature sets use the grammatical relation output of the RASP parser run over the written BNC. The **Rbasic** feature set conflates information about 25 grammatical relations; **Rmod** counts only prepositional, nominal and adjectival noun modification; **Rverb** counts only relations among subjects, objects and verbs; **Rconj** counts only conjunctions of nouns.

The feature vector for each target constituent counts its co-occurrences with the 10,000 words that most frequently appear in the co-occurrence relations of interest over the entire corpus. A feature vector for each compound was created by appending the vectors for its modifier and head, and these compound vectors were used for SVM learning. To model aspects of co-occurrence association that might be obscured by raw frequency, the log-likelihood ratio G^2 (Dunning, 1993) was also used to transform the feature space.

[5] Keerthi and Lin (2003) prove that the Gaussian kernel will always do as well as or better than the linear kernel for binary classification. For multiclass classification we use multiple binary classifiers with a shared set of parameters which may not be optimal for any single classifier.

	Raw		G^2	
	Accuracy	Macro	Accuracy	Macro
w5	52.60%	51.07%	51.35%	49.93%
w10	51.84%	50.32%	50.10%	48.60%
Rbasic	51.28%	49.92%	51.83%	50.26%
Rmod	51.35%	50.06%	48.51%	47.03%
Rverb	48.79%	47.13%	48.58%	47.07%
Rconj	**54.12%**	**52.44%**	**54.95%**	**53.42%**

Table 2: Classification results for word similarity

Micro- and macro-averaged performance figures are given in Table 2. The micro-averaged figure is calculated as the overall proportion of items that were classified correctly, whereas the macro-average is calculated as the average of the accuracy on each class and thus balances out any skew in the class distribution. In all cases macro-accuracy is lower than micro-accuracy; this is due to much better performance on the relations IN, INST, ACTOR and ABOUT than on BE and HAVE. This may be because those two relations are slightly rarer and hence provide less training data, or it may reflect a difference in the suitability of co-occurrence data for their classification. It is interesting that features derived only from conjunctions give the best performance; these features are the most sparse but appear to be of high quality. The information contained in conjunctions is conceptually very close to the WordNet-derived information frequently used in word-similarity based approaches to compound semantics, and the performance of these features is not far off the 56.76% accuracy (54.6% macro-average) reported for WordNet-based classification for the same dataset by Ó Séaghdha (2007).

5 Type Similarity

Type similarity is measured by identifying co-occurrences with each instance of the compound type in the corpus. In effect, we are treating compounds as single words and calculating their word similarity with each other. The same feature extraction methods were used as in the previous section. Classification results are given in Table 3.

This method performs very poorly. Sparsity is undoubtedly a factor: 513 of the 1,443 compounds oc-

	Accuracy	Macro
win5	28.62%	27.71%
win10	**30.01**%	**28.69**%
Rbasic	29.31%	28.22%
Rmod	26.54%	25.30%
Rverb	25.02%	23.96%
Rconj	24.60%	24.48%

Table 3: Classification results for type similarity

cur 5 times or fewer in the BNC and 186 occur just once. The sparser feature sets (**Rmod**, **Rverb** and **Rconj**) are all outperformed by the more dense ones. However, there is also a conceptual problem with type similarity, in that the context of a compound may contain information about the referent of the compound but is less likely to contain information about the implicit semantic relation. For example, the following compounds all encode different meanings but are likely to appear in similar contexts:

- John cut the bread with the *kitchen knife*.
- John cut the bread with the *steel knife*.
- John cut the bread with the *bread knife*.

6 Relation Similarity

6.1 Vector Space Kernels

The intuition underlying the use of relation similarity is that while the relation between the constituents of a compound may not be made explicit in the context of that compound, it may be described in other contexts where both constituents appear. For example, sentences containing both *bread* and *knife* may contain information about the typical interactions between their referents. To extract feature vectors for each constituent pair, we took the maximal context unit to be each sentence in which both constituents appear, and experimented with a range of refinements to that context definition. The resulting definitions are given below in order of intuitive richness, from measures based on word-counting to measures making use of the structure of the sentence's dependency parse graph.

allwords All words in the sentence are co-occurrence features. This context may be parameterised by specifying a limit on the window size to the left of the leftmost constituent and to the right of the rightmost constituent i.e., the words between the two constituents are always counted.

midwords All words between the constituents are counted.

allGRs All words in the sentence entering into a grammatical relation (with any other word) are counted. This context may be parameterised by specifying a limit on the length of the shortest path in the dependency graph from either of the target constituents to the feature word.

shortest path All words on the shortest dependency path between the two constituents are features. If there is no such path, no features are extracted.

path triples The shortest dependency path is decomposed into a set of triples and these triples are used as features. Each triple consists of a node on the shortest path (the triple's centre node) and two edges connecting that node with other nodes in the parse graph (not necessarily nodes on the path). To generate further triple features, one or both of the off-centre nodes is replaced by part(s) of speech. For example, the RASP dependency parse of *The knife cut the fresh bread* is:

```
(|ncsubj| |cut:3_VVD| |knife:2_NN1| _)
(|dobj| |cut:3_VVD| |bread:6_NN1|)
(|det| |bread:6_NN1| |the:4_AT|)
(|ncmod| _ |bread:6_NN1| |fresh:5_JJ|)
(|det| |knife:2_NN1| |The:1_AT|)
```

The derived set of features includes the triples

{the:A:det←knife:N←cut:V:ncsubj,
A:det←knife:N←cut:V:ncsubj,
the:A:det←knife:N←V:ncsubj,
A:det←knife:N←V:ncsubj,
knife:N:ncsubj←cut:V→bread:N:dobj,
N:ncsubj←cut:V→bread:N:dobj,
knife:N:ncsubj←cut:V→N:dobj,
N:ncsubj←cut:V→N:dobj,...}

(The ← and → arrows indicate the direction of the head-modifier dependency)

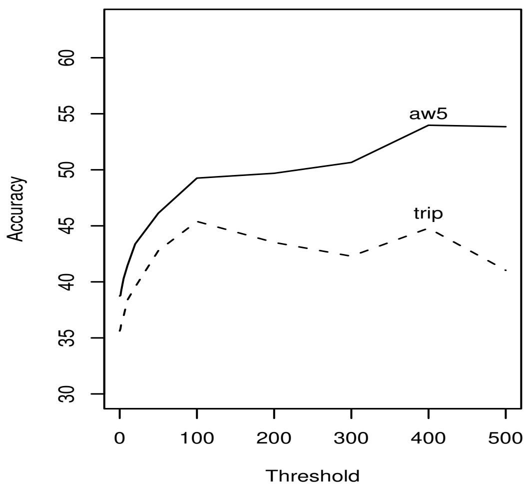

Figure 1: Effect of BNC frequency on test item accuracy for the **allwords5** and **triples** contexts

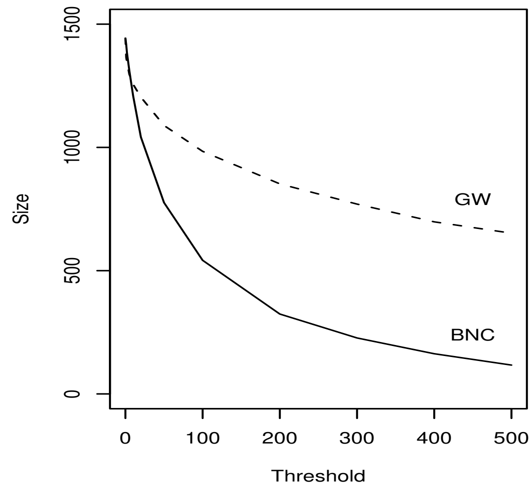

Figure 2: Effect of corpus frequency on dataset size for the BNC and Gigaword-derived corpus

Table 4 presents results for these contexts; in the case of parameterisable contexts the best-performing parameter setting is presented. We are currently unable to present results for the path-based contexts using the Gigaword corpus. It is clear from the accuracy figures that we have not matched the performance of the word similarity approach. The best-performing single context definition is **allwords** with a window parameter of 5, which yields accuracy of 38.74% (36.78% macro-average). We can combine the contributions of two contexts by generating a new kernel that is the sum of the linear kernels for the individual contexts;[6] the sum of **allwords5** and **triples** achieves the best performance with 42.34% (40.20% macro-average).

It might be expected that the richer context definitions provide sparser but more precise information, and that their relative performance might improve when only frequently observed word pairs are to be classified. However, thresholding inclusion in the test set on corpus frequency belies that expectation; as the threshold increases and the testing data contains only more frequent pairs, all contexts show improved performance but the effect is strongest for the **allwords** and **midwords** contexts. Figure 1 shows threshold-accuracy curves for two representative contexts (the macro-accuracy curves are similar).

For all frequency thresholds above 6, the number of noun pairs with above-threshold corpus frequency is greater for the Gigaword corpus than for the BNC, and this effect is amplified with increasing threshold (see Figure 2). However, this difference in sparsity does not always induce an improvement in performance, but nor does the difference in corpus type consistently favour the BNC.

	BNC		Gigaword	
	Accuracy	Macro	Accuracy	Macro
aw	35.97%	33.39%	34.58%	32.62%
aw5	**38.74%**	**36.78%**	37.28%	35.25%
mw	32.29%	30.38%	36.24%	34.25%
agr	35.34%	33.40%	35.34%	33.34%
agr2	36.73%	34.81%	37.28%	35.59%
sp	33.54%	31.51%		
trip	35.62%	34.39%		
aw5+ trip	**42.34%**	**40.20%**		

Table 4: Classification results for relation similarity

[6]The summed kernel function value for a pair of items is simply the sum of the two kernel functions' values for the pair, i.e.:

$$K_{sum}(x, y) = K_1(\phi_1(x), \phi_1(y)) + K_2(\phi_2(x), \phi_2(y))$$

where ϕ_1, ϕ_2 are the context representations used by the two kernels. A detailed study of kernel combination is presented by Joachims et al. (2001).

6.2 String Kernels

The classification techniques described in the previous subsection represent the relational context for each word pair as a co-occurrence vector in an inner product space and compute the similarity between two pairs as a function of their vector representations. A different kind of similarity measure is provided by *string kernels*, which count the number of subsequences shared by two strings. This class of kernel function implicitly calculates an inner product in a feature space indexed by all possible subsequences (possibly restricted by length or contiguity), but the feature vectors are not explicitly represented. This approach affords our notion of context an increase in richness (features can be sequences of length ≥ 1) without incurring the computational cost of the exponential growth in the dimension of our feature space. A particularly flexible string kernel is the gap-weighted kernel described by Lodhi et al. (2002), which allows the subsequences to be non-contiguous but penalises the contribution of each subsequence to the kernel value according to the number of items occurring between the start and end of the subsequence, including those that do not belong to the subsequence (the "gaps").

The kernel is defined as follows. Let s and t be two strings of words belonging to a vocabulary Σ. A subsequence u of s is defined by a sequence of indices $\mathbf{i} = (i_1, \ldots, i_{|u|})$ such that $1 \leq i_1 < \ldots < i_{|u|} \leq |s|$, where s is the length of s. Let $l(\mathbf{i}) = i_{|u|} - i_1 + 1$ be the length of the subsequence in s. For example, if s is the string "cut the bread with the knife" and u is the subsequence "cut with" indexed by $\mathbf{i}$ then $l(\mathbf{i}) = 4$. λ is a decay parameter between 0 and 1. The gap-weighted kernel value for subsequences of length n of strings s and t is given by

$$K_{S_n}(s,t) = \sum_{u \in \Sigma^n} \sum_{\mathbf{i},\mathbf{j}: s[\mathbf{i}]=u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}$$

Directly computing this function would be intractable, as the sum is over all $|\Sigma|^n$ possible subsequences of length n; however, Lodhi et al. (2002) present an efficient dynamic programming algorithm that can evaluate the kernel in $O(n|s||t|)$ time. Those authors' application of string kernels to text categorisation counts sequences of characters, but it is generally more suitable for NLP applications to use sequences of words (Cancedda et al., 2003).

This kernel calculates a similarity score for a pair of strings, but for context-based compound classification we are interested in the similarity between two *sets* of strings. We therefore define a *context kernel*, which sums the kernel scores for each pair of strings from the two context sets C_1, C_2 and normalises them by the number of pairs contributing to the sum:

$$K_{C_n}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{s \in C_1, t \in C_2} K_{S_n}(s,t)$$

That this is a valid kernel (i.e., defines an inner product in some induced vector space) can be proven using the definition of the *derived subsets kernel* in Shawe-Taylor and Cristianini (2004, p. 317). In our experiments we further normalise the kernel to ensure that $K_{C_n}(C_1, C_2) = 1$ if and only if $C_1 = C_2$.

To generate the context set for a given word pair, we extract a string from every sentence in the BNC where the pair of words occurs no more than eight words apart. On the hypothesis that the context between the target words was most important and to avoid the computational cost incurred by long strings, we only use this middle context. To facilitate generalisations over subsequences, the compound head is replaced by a marker HEAD and the modifier is replaced by a marker MOD. Word pairs for which no context strings were extracted (i.e., pairs which only occur as compounds in the corpus) are represented by a dummy string that matches no other. The value of λ is set to 0.5 as in Cancedda et al. (2003). Table 5 presents results for the context kernels with subsequence lengths 1,2,3 as well as the kernel sum of these three kernels. These kernels perform better than the relational vector space kernels, with the exception of the summed **allwords5 + triples** kernel.

7 Combining Contexts

We can use the method of kernel summation to combine information from different context types. If our intuition is correct that type and relation similarity provide different "views" of the same semantic relation, we would expect their combination to give better results than either taken alone. This is also suggested by the observation that the different context

	Accuracy	Macro
$n = 1$	15.94%	19.88%
$n = 2$	39.09%	37.23%
$n = 3$	39.29%	**39.29%**
$\Sigma_{1,2,3}$	**40.61%**	38.53%

Table 5: Classification results for gap-weighted string kernels with subsequence lengths 1,2,3 and the kernel sum of these kernels

	Accuracy	Macro
Rconj-G^2 + **aw5**	54.95%	53.50%
Rconj-G^2 + **triples**	56.20%	54.54%
Rconj-G^2 + **aw5** + **triples**	55.86%	54.13%
Rconj-G^2 + K_{C_2}	56.48%	54.89%
Rconj-G^2 + K_{C_Σ}	**56.55%**	**54.96%**

Table 6: Classification results for context combinations

types favour different relations: the summed string kernel is the best at identifying IN relations (70.45% precision, 46.67% recall), but Rconj-G^2 is best at identifying all others. This intuition is confirmed by our experiments, the results of which appear in Table 6. The best performance of 56.55% accuracy (54.96% macro-average) is attained by the combination of the G^2-transformed **Rconj** word similarity kernel and the summed string kernel K_{C_Σ}. We note that this result, using only information extracted from the BNC, compares favourably with the 56.76% accuracy (54.60% macro-average) results described by Ó Séaghdha (2007) for a WordNet-based method. The combination of **Rconj**-G^2 and **triples** is also competitive, demonstrating that a less flexible learning algorithm (the linear kernel) can perform well if it has access to a richer source of information (dependency paths).

8 Comparison with Prior Work

Previous work on compound semantics has tended to concentrate on either word or relation similarity. Approaches based on word similarity generally use information extracted from WordNet. For example, Girju et al. (2005) train SVM classifiers on hypernymy features for each constituent. Their best reported accuracy with an equivalent level of supervision to our work is 54.2%; they then improve performance by adding a significant amount of manually-annotated semantic information to the data, as does Girju (2006) in a multilingual context. It is difficult to make any conclusive comparison with these results due to fundamental differences in datasets and classification schemes.

Approaches based on relational similarity often use relative frequencies of fixed lexical sequences estimated from massive corpora. Lapata and Keller (2004) use Web counts for phrases *Noun P Noun* where *P* belongs to a predefined set of prepositions. This unsupervised approach gives state-of-the-art results on the assignment of prepositional paraphrases, but cannot be applied to deep semantic relations which cannot be directly identified in text. Turney and Littman (2005) search for phrases *Noun R Noun* where *R* is one of 64 "joining words". Turney (2006) presents a more flexible framework in which automatically identified n-gram features replace fixed unigrams and additional word pairs are generated by considering synonyms, but this method still requires a Web-magnitude corpus and a very large amount of computational time and storage space. The latter paper reports accuracy of 58.0% (55.9% macro-average), which remains the highest reported figure for corpus-based approaches and demonstrates that relational similarity can perform well given sufficient resources.

We are not aware of previous work that compares the effectiveness of different classes of context for compound interpretation, nor of work that investigates the utility of different corpora. We have also described the first application of string kernels to the compound task, though gap-weighted kernels have been used successfully for related tasks such as word sense disambiguation (Gliozzo et al., 2005) and relation extraction (Bunescu and Mooney, 2005).

9 Conclusion and Future Work

We have defined four kinds of co-occurrence contexts for compound interpretation and demonstrated that word similarity outperforms a range of relation contexts using information derived from the British National Corpus. Our experiments with the English Gigaword Corpus indicate that more data is not always better, and that large newswire corpora may not be ideally suited to general relation-based tasks.

On the other hand it might be expected to be very useful for disambiguating relations more typical of news stories (such as *tax cut*, *rail strike*).

Future research directions include developing more sophisticated context kernels. Cancedda et al. (2003) present a number of potentially useful refinements of the gap-weighted string kernel, including "soft matching" and differential values of λ for different words or word classes. We intend to combine the benefits of string kernels with the linguistic richness of syntactic parses by computing subsequence kernels on dependency paths. We have also begun to experiment with the tree kernels of Moschitti (2006), but are not yet in a position to report results. As mentioned in Section 2, we also intend to investigate the potential contribution of the sentential contexts that contain the compound tokens to be classified (token similarity).

While the BNC has many desirable properties, it may also be fruitful to investigate the utility of a large encyclopaedic corpus such as Wikipedia, which may be more explicit in its description of relations between real-world entities than typical text corpora. Wikipedia has shown promise as a resource for measuring word similarity (Strube and Ponzetto, 2006) and relation similarity (Suchanek et al. (2006)).

References

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. 2005. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems*.

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

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL-04*.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

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. 2006. Out-of-context noun phrase semantic interpretation with cross-linguistic evidence. In *Proceedings of CIKM-06*.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL-05*.

David Graff, 2003. *English Gigaword*. Linguistic Data Consortium, Philadelphia.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. In *Proceedings of ICML-01*.

S. Sathiya Keerthi and Chih-Jen Lin. 2003. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15:1667–1689.

Mirella Lapata and Frank Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proceedings of HLT-NAACL-04*.

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

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

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML-06*.

Sebastian Padó and Mirella Lapata. 2003. Constructing semantic space models from parsed corpora. In *Proceedings of ACL-03*.

Diarmuid Ó Séaghdha. 2007. Annotating and learning compound noun semantics. In *Proceedings of the ACL-07 Student Research Workshop*.

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

Michael Strube and Simone Paolo Ponzetto. 2006. WikiRelate! computing semantic relatedness using Wikipedia. In *Proceedings of AAAI-06*.

Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. LEILA: Learning to extract information by linguistic analysis. In *Proceedings of the ACL-06 Workshop on Ontology Learning and Population*.

Peter D. Turney and Michael L. Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1–3):251–278.

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