Unsupervised Lexical Acquisition: Proceedings of the Workshop of the ACL Special Interest Group on the Lexicon (SIGLEX), Philadelphia, July 2002, pp. 59-66. Association for Computational Linguistics.

Improvements in Automatic Thesaurus Extraction

James R. Curran and **Marc Moens**
Institute for Communicating and Collaborative Systems
University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW
United Kingdom
{jamesc,marc}@cogsci.ed.ac.uk

Abstract

The use of semantic resources is common in modern NLP systems, but methods to extract lexical semantics have only recently begun to perform well enough for practical use. We evaluate existing and new similarity metrics for thesaurus extraction, and experiment with the trade-off between extraction performance and efficiency. We propose an approximation algorithm, based on *canonical attributes* and coarse- and fine-grained matching, that reduces the time complexity and execution time of thesaurus extraction with only a marginal performance penalty.

1 Introduction

Thesauri have traditionally been used in information retrieval tasks to expand words in queries with synonymous terms (e.g. Ruge, (1997)). Since the development of WordNet (Fellbaum, 1998) and large electronic thesauri, information from semantic resources is regularly leveraged to solve NLP problems. These tasks include collocation discovery (Pearce, 2001), smoothing and model estimation (Brown et al., 1992; Clark and Weir, 2001) and text classification (Baker and McCallum, 1998).

Unfortunately, thesauri are expensive and time-consuming to create manually, and tend to suffer from problems of bias, inconsistency, and limited coverage. In addition, thesaurus compilers cannot keep up with constantly evolving language use and cannot afford to build new thesauri for the many subdomains that NLP techniques are being applied to. There is a clear need for methods to extract thesauri automatically or tools that assist in the manual creation and updating of these semantic resources.

Much of the existing work on thesaurus extraction and word clustering is based on the observation that related terms will appear in *similar* contexts. These systems differ primarily in their definition of "context" and the way they calculate similarity from the contexts each term appears in.

Most systems extract co-occurrence and syntactic information from the words surrounding the target term, which is then converted into a vector-space representation of the contexts that each target term appears in (Pereira et al., 1993; Ruge, 1997; Lin, 1998b). Other systems take the whole document as the context and consider term co-occurrence at the document level (Crouch, 1988; Sanderson and Croft, 1999). Once these contexts have been defined, these systems then use clustering or nearest neighbour methods to find similar terms.

Alternatively, some systems are based on the observation that related terms appear *together* in particular contexts. These systems extract related terms directly by recognising linguistic patterns (e.g. *X, Y and other Zs*) which link synonyms and hyponyms (Hearst, 1992; Caraballo, 1999).

Our previous work (Curran and Moens, 2002) has evaluated thesaurus extraction performance and efficiency using several different context models. In this paper, we evaluate some existing similarity metrics and propose and motivate a new metric which outperforms the existing metrics. We also present an approximation algorithm that bounds the time complexity of pairwise thesaurus extraction. This results in a significant reduction in runtime with only a marginal performance penalty in our experiments.

2 Automatic Thesaurus Extraction

Vector-space thesaurus extraction systems can be separated into two components. The first component extracts the contexts from raw text and compiles them into a statistical description of the contexts each potential thesaurus term appears in. Some systems define the context as a window of words surrounding each thesaurus term (McDonald, 2000). Many systems extract grammatical relations using either a broad coverage parser (Lin, 1998a) or shallow statistical tools (Grefenstette, 1994; Curran and Moens, 2002). Our experiments use a shallow relation extractor based on (Grefenstette, 1994).

We define a *context relation* instance as a tuple (w, r, w') where w is the thesaurus term, which occurs in some grammatical relation r with another word w' in the sentence. We refer to the tuple (r, w') as an *attribute* of w. For example, the tuple `(dog, direct-obj, walk)` indicates that the term `dog` was the direct object of the verb `walk`.

Our relation extractor begins with a Naïve Bayes POS tagger and chunker. After the raw text has been tagged and chunked, noun phrases separated by prepositions and conjunctions are concatenated, and the relation extracting algorithm is run over each sentence. This consists of four passes over the sentence, associating each noun with the modifiers and verbs from the syntactic contexts they appear in:

1. nouns with pre-modifiers (left to right)
2. nouns with post-modifiers (right to left)
3. verbs with subjects/objects (right to left)
4. verbs with subjects/objects (left to right)

This results in tuples representing the contexts:

1. term is the subject of a verb
2. term is the (direct/indirect) object of a verb
3. term is modified by a noun or adjective
4. term is modified by a prepositional phrase

The relation tuple is then converted to root form using the Sussex morphological analyser (Minnen et al., 2000) and the POS tags are removed. The relations for each term are collected together and counted, producing a context vector of attributes and their frequencies in the corpus. Figure 1 shows some example attributes for `idea`.

```
(adjective, good) 2005
(adjective, faintest) 89
(direct-obj, have) 1836
(indirect-obj, toy) 74
(adjective, preconceived) 42
(adjective, foggiest) 15
```

Figure 1: Example attributes of the noun `idea`

The second system component performs nearest-neighbour or cluster analysis to determine which terms are similar based on their context vectors. Both methods require a function that calculates the similarity between context vectors. For experimental analysis we have decomposed this function into *measure* and *weight* functions. The *measure* function calculates the similarity between two weighted context vectors and the *weight* function calculates a weight from the raw frequency information for each context relation. The primary experiments in this paper evaluate the performance of various existing and new measure and weight functions, which are described in the next section.

The simplest algorithm for thesaurus extraction is nearest-neighbour comparison, which involves pairwise vector comparison of the target with every extracted term. Given n terms and up to m attributes for each term, the asymptotic time complexity of nearest-neighbour thesaurus extraction is $O(n^2m)$. This is very expensive with even a moderate vocabulary and small attribute vectors. The number of terms can be reduced by introducing a minimum cutoff that ignores potential synonyms with a frequency less than the cutoff, which for our experiments was 5. Section 5 reports on the trade-off between the minimum cutoff and execution time.

3 Experiments

Early experiments in thesaurus extraction (Grefenstette, 1994) suffered from the limited size of available corpora, but more recent experiments have used much larger corpora with greater success (Lin, 1998a). For these experiments we ran our relation extractor over the British National Corpus (BNC) consisting of 114 million words in 6.2 million sentences. The POS tagging and chunking took 159 minutes, and the relation extraction took an addi-

SETCOSINE	$\frac{\lvert(w_m,*,*)\cap(w_n,*,*)\rvert}{\sqrt{\lvert(w_m,*,*)\rvert\times\lvert(w_n,*,*)\rvert}}$
COSINE	$\frac{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})\times\mathrm{wgt}(w_n,*_r,*_{w'})}{\sqrt{\sum \mathrm{wgt}(w_m,*,*)^2\times\sum \mathrm{wgt}(w_n,*,*)^2}}$
SETDICE	$\frac{2\lvert(w_m,*,*)\cap(w_n,*,*)\rvert}{\lvert(w_m,*,*)\rvert+\lvert(w_n,*,*)\rvert}$
DICE	$\frac{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})\times\mathrm{wgt}(w_n,*_r,*_{w'})}{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})+\mathrm{wgt}(w_n,*_r,*_{w'})}$
DICE†	$\frac{2\sum_{(r,w')} \min(\mathrm{wgt}(w_m,*_r,*_{w'}),\mathrm{wgt}(w_n,*_r,*_{w'}))}{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})+\mathrm{wgt}(w_n,*_r,*_{w'})}$
SETJACCARD	$\frac{\lvert(w_m,*,*)\cap(w_n,*,*)\rvert}{\lvert(w_m,*,*)\cup(w_n,*,*)\rvert}$
JACCARD	$\frac{\sum_{(r,w')} \min(\mathrm{wgt}(w_m,*_r,*_{w'}),\mathrm{wgt}(w_n,*_r,*_{w'}))}{\sum_{(r,w')} \max(\mathrm{wgt}(w_m,*_r,*_{w'}),\mathrm{wgt}(w_n,*_r,*_{w'}))}$
JACCARD†	$\frac{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})\times\mathrm{wgt}(w_n,*_r,*_{w'})}{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})+\mathrm{wgt}(w_n,*_r,*_{w'})}$
LIN	$\frac{\sum_{(r,w')} \mathrm{wgt}(w_m,*_r,*_{w'})+\mathrm{wgt}(w_n,*_r,*_{w'})}{\sum \mathrm{wgt}(w_m,*,*)+\sum \mathrm{wgt}(w_n,*,*)}$

Table 1: Measure functions evaluated

tional 7.5 minutes. The resultant representation contained a total of 28 million relation occurrences over 10 million different relations.

We describe the functions evaluated in these experiments using an extension of the asterisk notation used by Lin (1998a), where an asterisk indicates a set ranging over all existing values of that variable. For example, the set of attributes of the term w is:

$$(w, *, *) \equiv \{(r, w') \mid \exists (w, r, w')\}$$

For convenience, we further extend the notation for weighted attribute vectors. A subscripted asterisk indicates that the variables are bound together:

$$\sum_{(r,w')} \mathrm{wgt}(w_m, *_r, *_{w'}) \times \mathrm{wgt}(w_n, *_r, *_{w'})$$

which is a notational abbreviation of:

$$\sum_{(r,w')\in(w_m,*,*)\cap(w_n,*,*)} \mathrm{wgt}(w_m, r, w') \times \mathrm{wgt}(w_n, r, w')$$

For weight functions we use similar notation:

$$\begin{aligned} f(w,*,*) &\equiv \sum_{(r,w')\in(w,*,*)} f(w,r,w') \\ n(w,*,*) &\equiv |(w,*,*)| \\ N_w &\equiv |\{w \mid \exists(w,*,*) \neq \emptyset\}| \end{aligned}$$

Table 1 defines the measure functions evaluated in these experiments. The simplest measure functions (prefix SET) use the attribute set model from IR and are taken from Manning and Schütze (1999), pp. 299. When these are used with weighted attributes, if the weight is greater than zero, then it is considered in the set. Other measures, such as LIN and JACCARD have previously been used for thesaurus extraction (Lin, 1998a; Grefenstette, 1994). Finally, we have generalised some set measures using similar reasoning to Grefenstette (1994). Alternative generalisations are marked with a dagger.

IDENTITY	1.0
CHI2	*cf. Manning and Schütze (1999)*
LR	*cf. Manning and Schütze (1999)*
LIN98A	$\log(\frac{f(w,r,w')f(*,r,*)}{f(*,r,w')f(w,r,*)})$
LIN98B	$-\log(\frac{n(*,r,w')}{N_w})$
DICE	$\frac{2p(w,r,w')}{p(w,*,*)+p(*,r,w')}$
GREF94	$\frac{\log_2(f(w,r,w')+1)}{\log_2(n(*,r,w')+1)}$
MI	$\log(\frac{p(w,r,w')}{p(w,*,*)p(*,r,w')})$
TTEST	$\frac{p(w,r,w')-p(*,r,w')p(w,*,*)}{\sqrt{p(*,r,w')p(w,*,*)}}$

Table 2: Weight functions evaluated

These experiments also cover a range of weight functions as defined in Table 2. The weight functions LIN98A, LIN98B, and GREF94 are taken from existing systems (Lin, 1998a; Lin, 1998b; Grefenstette, 1994). Our proposed weight functions are motivated by our intuition that highly predictive attributes are strong collocations with their terms. Thus, we have implemented many of the statistics described in the *Collocations* chapter of Manning and Schütze (1999), including the T-Test, χ^2-Test, Likelihood Ratio, and Mutual Information. Some functions (suffix LOG) have an extra $\log_2(f(w, r, w') + 1)$ factor to promote the influence of higher frequency attributes.

4 Evaluation

For the purposes of evaluation, we selected 70 single-word noun terms for thesaurus extraction. To avoid sample bias, the words were randomly selected from WordNet such that they covered a range of values for the following word properties:

Word	PTB Rank	PTB #	BNC #	Reuters #	Macquarie #	WordNet #	Min / Max	WordNet subtree roots
`company`	38	4076	52779	456580	8	9	3 / 6	entity, group, state
`interest`	138	919	37454	146043	12	12	3 / 8	abs., act, group, poss., state
`problem`	418	622	56361	63333	4	3	3 / 7	abs., psych., state
`change`	681	406	35641	55081	8	10	2 / 12	abs., act, entity, event, phenom.
`idea`	1227	134	32754	13527	10	5	3 / 7	entity, psych.
`radio`	2278	59	9046	20913	2	3	6 / 8	entity
`star`	5130	29	8301	6586	11	7	4 / 8	abs., entity
`knowledge`	5197	19	14580	2813	3	1	1 / 1	psych.
`pants`	13264	5	429	282	3	2	6 / 9	entity
`tightness`	30817	1	119	2020	5	3	4 / 5	abs., state

Table 3: Examples of the 70 thesaurus evaluation terms

frequency Penn Treebank and BNC frequencies;

number of senses WordNet and Macquarie senses;

specificity depth in the WordNet hierarchy;

concreteness distribution across WordNet subtrees.

Table 3 lists some example terms with frequency and frequency rank data from the PTB, BNC and REUTERS, as well as the number of senses in WordNet and Macquarie, and their maximum and minimum depth in the WordNet hierarchy. For each term we extracted a thesaurus entry with 200 potential synonyms and their similarity scores.

The simplest method of evaluation is direct comparison of the extracted thesaurus with a manually-created gold standard (Grefenstette, 1994). However, on small corpora, rare direct matches provide limited information for evaluation, and thesaurus coverage is a problem. Our evaluation uses a combination of three electronic thesauri: the Macquarie (Bernard, 1990), Roget's (Roget, 1911) and Moby (Ward, 1996) thesauri. Roget's and Macquarie are topic ordered and the Moby thesaurus is head ordered. As the extracted thesauri do not distinguish between senses, we transform Roget's and Macquarie into head ordered format by conflating the sense sets containing each term. For the 70 terms we create a gold standard from the union of the synonyms from the three thesauri.

With this gold standard in place, it is possible to use precision and recall measures to evaluate the quality of the extracted thesaurus. To help overcome the problems of direct comparisons we use several measures of system performance: direct matches (DIRECT), inverse rank (INVR), and precision of the top n synonyms (P(n)), for n = 1, 5 and 10.

Measure	DIRECT	P(1)	P(5)	P(10)	INVR
SETCOSINE	1276	14%	15%	15%	0.76
SETDICE	1496	63%	44%	34%	1.69
SETJACCARD	1458	59%	43%	34%	1.63
COSINE	1276	14%	15%	15%	0.76
DICE	1536	19%	20%	20%	0.97
DICE†	1916	76%	52%	45%	2.10
JACCARD	1916	76%	52%	45%	2.10
JACCARD†	1745	40%	30%	28%	1.36
LIN	1826	60%	46%	40%	1.85

Table 4: Evaluation of measure functions

INVR is the sum of the inverse rank of each matching synonym, e.g. matching synonyms at ranks 3, 5 and 28 give an inverse rank score of $\frac{1}{3}+\frac{1}{5}+\frac{1}{28}$, and with at most 200 synonyms, the maximum INVR score is 5.878. Precision of the top n is the percentage of matching synonyms in the top n extracted synonyms. There are a total of 23207 synonyms for the 70 terms in the gold standard. Each measure is averaged over the extracted synonym lists for all 70 thesaurus terms.

5 Results

For computational practicality, we assume that the performance behaviour of measure and weight functions are independent of each other. Therefore, we have evaluated the weight functions using the JACCARD measure, and evaluated the measure functions using the TTEST weight because they produced the best results in our previous experiments.

Table 4 presents the results of evaluating the measure functions. The best performance across all measures was shared by JACCARD and DICE†, which produced identical results for the 70 words. DICE† is easier to compute and is thus the preferred measure function.

Table 5 presents the results of evaluating the

Weight	DIRECT	P(1)	P(5)	P(10)	INVR
CHI2	1623	33%	27%	26%	1.24
DICE	1480	61%	45%	34%	1.70
DICELOG	1498	67%	45%	35%	1.73
GREF94	1258	54%	38%	29%	1.46
IDENTITY	1228	46%	34%	29%	1.33
LR	1510	53%	39%	32%	1.58
LIN98A	1735	73%	50%	42%	1.96
LIN98B	1271	47%	34%	30%	1.37
MI	1736	66%	49%	42%	1.92
MILOG	1841	71%	52%	43%	2.05
TTEST	1916	76%	52%	45%	2.10
TTESTLOG	1865	70%	49%	41%	1.99

Table 5: Evaluation of bounded weight functions

Weight	DIRECT	P(1)	P(5)	P(10)	INVR
MI$^{\pm}$	1511	59%	44%	39%	1.74
MILOG$^{\pm}$	1566	61%	46%	41%	1.84
TTEST$^{\pm}$	1670	67%	50%	43%	1.96
TTESTLOG$^{\pm}$	1532	63%	50%	42%	1.89

Table 6: Evaluation of unbounded weight functions

weight functions. Here TTEST significantly outperformed the other weight functions, which supports our intuition that good context descriptors are also strong collocates of the term. Surprisingly, the other collocation discovery functions did not perform as well, even though TTEST is not the most favoured for collocation discovery because of its behaviour at low frequency counts.

One difficulty with weight functions involving logarithms or differences is that they can be negative. The results in Table 6 show that weight functions that are not bounded below by zero do not perform as well on thesaurus extraction. However, unbounded weights do produce interesting and unexpected results: they tend to return misspellings of the term and synonyms, abbreviations and lower frequency synonyms. For instance, TTEST$^{\pm}$ returned `Co`, `Co.` and `PLC` for `company`, but they do not appear in the synonyms extracted with TTEST. The unbounded weights also extracted more hyponyms, such as corporation names for `company`, including `Kodak` and `Exxon`. Finally unbounded weights tended to promote the rankings of synonyms from minority senses because the frequent senses are demoted by negative weights. For example, TTEST$^{\pm}$ returned `writings`, `painting`, `fieldwork`, `essay` and `masterpiece` as the best synonyms for `work`, whereas TTEST returned `study`, `research`, `job`, `activity` and `life`.

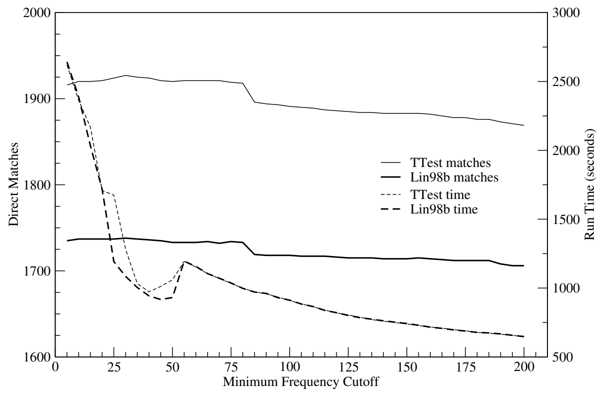

Figure 2: Performance against minimum cutoff

Introducing a minimum cutoff that ignores low frequency potential synonyms can eliminate many unnecessary comparisons. Figure 2 presents both the performance of the system using direct match evaluation (left axis) and execution times (right axis) for increasing cutoffs. This test was performed using JACCARD and the TTEST and LIN98A weight functions. The first feature of note is that as we increase the minimum cutoff to 30, the direct match results improve for TTEST, which is probably a result of the TTEST weakness on low frequency counts. Initially, the execution time is rapidly reduced by small increments of the minimum cutoff. This is because Zipf's law applies to relations, and so by small increments of the cutoff we eliminate many terms from the tail of the distribution. There are only 29,737 terms when the cutoff is 30; 88,926 terms when the cutoff is 5; and 246,067 without a cutoff, and because the extraction algorithm is $O(n^2 m)$, this results in significant efficiency gains. Since extracting only 70 thesaurus terms takes about 43 minutes with a minimum cutoff of 5, the efficiency/performance trade-off is particularly important from the perspective of implementing a practical extraction system.

6 Efficiency

Even with a minimum cutoff of 30 as a reasonable compromise between speed and accuracy, extracting a thesaurus for 70 terms takes approximately 20 minutes. If we want to extract a complete thesaurus for 29,737 terms left after the cutoff has been applied, it would take approximately one full week of processing. Given that the size of the training corpus could be much larger (cf.

Curran and Moens (2002)), which would increase both number of attributes for each term and the total number of terms above the minimum cutoff, this is not nearly fast enough. The problem is that the time complexity of thesaurus extraction is not practically scalable to significantly larger corpora.

Although the minimum cutoff helps by reducing n to a reasonably small value, it does not constrain m in any way. In fact, using a cutoff increases the average value of m across the terms because it removes low frequency terms with few attributes. For instance, the frequent `company` appears in 11360 grammatical relations, with a total frequency of 69240 occurrences, whereas the infrequent `pants` appears in only 401 relations with a total frequency of 655 occurrences.

The problem is that for every comparison, the algorithm must examine the length of both attribute vectors. Grefenstette (1994) uses bit signatures to test for shared attributes, but because of the high frequency of the most common attributes, this does not skip many comparisons. Our system keeps track of the sum of the remaining vector which is a significant optimisation, but comes at the cost of increased representation size. However, what is needed is some algorithmic reduction that bounds the number of full $O(m)$ vector comparisons performed.

7 Approximation Algorithm

One way of bounding the complexity is to perform an approximate comparison first. If the approximation returns a positive result, then the algorithm performs the full comparison. We can do this by introducing another, much shorter vector of *canonical* attributes, with a bounded length k. If our approximate comparison returns at most p positive results for each term, then the time complexity becomes $O(n^2k + npm)$, which, since k is constant, is $O(n^2 + npm)$. So as long as we find an approximation function and vector such that $p \ll n$, the system will run much faster and be much more scalable in m, the number of attributes. However, $p \ll n$ implies that we are discarding a very large number of potential matches and so there will be a performance penalty. This trade-off is governed by the number of the canonical attributes and how representative they are of the full attribute vector, and thus the term itself. It is also dependent on the functions used to compare the canonical attribute vectors.

```
(adjective, smarty) 3 0.0524
(direct-obj, pee) 3 0.0443
(noun-mod, loon) 5 0.0437
(direct-obj, wet) 14 0.0370
(direct-obj, scare) 10 0.0263
(adjective, jogging) 5 0.0246
(indirect-obj, piss) 4 0.0215
(noun-mod, ski) 14 0.0201
```

Figure 3: The top weighted attributes of `pants`

```
(direct-obj, wet) 14 0.0370
(direct-obj, scare) 10 0.0263
(direct-obj, wear) 17 0.0071
(direct-obj, keep) 7 0.0016
(direct-obj, get) 5 0.0004
```

Figure 4: Canonical attributes for `pants`

The canonical vector must contain attributes that best describe the thesaurus term in a bounded number of entries. The obvious first choice is the most strongly weighted attributes from the full vector. Figure 3 shows some of the most strongly weighted attributes for `pants` with their frequencies and weights. However, these attributes, although strongly correlated with `pants`, are in fact too specific and idiomatic to be a good summary, because there are very few other words with similar canonical attributes. For example, `(adjective, smarty)` only appears with two other terms (`bun` and `number`) in the entire corpus. The heuristic is so aggressive that too few positive approximate matches result.

To alleviate this problem we filter the attributes so that only strongly weighted `subject`, `direct-obj` and `indirect-obj` relations are included in the canonical vectors. This is because in general they constrain the terms more and partake in fewer idiomatic collocations with the terms. So the general principle is the most descriptive verb relations constrain the search for possible synonyms, and the other modifiers provide finer grain distinctions used to rank possible synonyms. Figure 4 shows the 5 canonical attributes for `pants`. This canonical vector is a better general description of the term `pants`, since similar terms are likely to appear as the direct object of `wear`, even though it still contains the idiomatic attributes `(direct-obj, wet)` and `(direct-obj, scare)`.

One final difficulty this example shows is that at-

Word	DIRECT	BIG / MAX	P(1)	P(5)	P(10)	INVR	BIG / MAX
`company`	27	110 / 355	100 %	80 %	60 %	2.60	2.71 / 6.45
`interest`	64	232 / 730	100 %	80 %	70 %	3.19	3.45 / 7.17
`problem`	25	82 / 250	100 %	60 %	50 %	2.46	2.52 / 6.10
`change`	31	104 / 544	100 %	60 %	40 %	2.35	2.44 / 6.88
`idea`	59	170 / 434	100 %	100 %	80 %	3.67	3.87 / 6.65
`radio`	19	45 / 177	100 %	60 %	60 %	2.31	2.35 / 5.76
`star`	31	141 / 569	100 %	60 %	60 %	2.36	2.49 / 6.92
`knowledge`	26	56 / 151	100 %	80 %	70 %	2.50	2.55 / 5.60
`pants`	12	13 / 222	100 %	80 %	50 %	2.40	2.40 / 5.98
`tightness`	3	3 / 152	0 %	0 %	0 %	0.03	0.03 / 5.60
Average (over 70)	26	86 / 332	76 %	52 %	44 %	2.08	2.17 / 6.13

Table 7: Example performance using techniques described in this paper

tributes like `(direct-obj, get)` are not informative. We know this because `(direct-obj, get)` appears with 8769 different terms, which means the algorithm may perform a large number of unnecessary full comparisons since `(direct-obj, get)` could be a canonical attribute for many terms. To avoid this problem, we apply a maximum cutoff on the number of terms the attribute appears with.

With limited experimentation, we have found that TTESTLOG is the best weight function for selecting canonical attributes. This may be because the extra $\log_2(f(w, r, w') + 1)$ factor encodes the desired bias towards relatively frequent canonical attributes. If a canonical attribute is shared by the two terms, then our algorithm performs the full comparison.

Figure 5 shows system performance and speed, as canonical vector size is increased, with the maximum cutoff at 4000, 8000, and 10,000. As an example, with a maximum cutoff of 10,000 and a canonical vector size of 70, the total DIRECT score of 1841 represents a 3.9% performance penalty over full extraction, for an 89% reduction in execution time. Table 7 presents the example term results using the techniques we have described: JACCARD measure and TTEST weight functions; minimum cutoff of 30; and approximation algorithm with canonical vector size of 100 with TTESTLOG weighting. The BIG columns show the previous measure results if we returned 10,000 synonyms, and MAX gives the results for a comparison of the gold standard against itself.

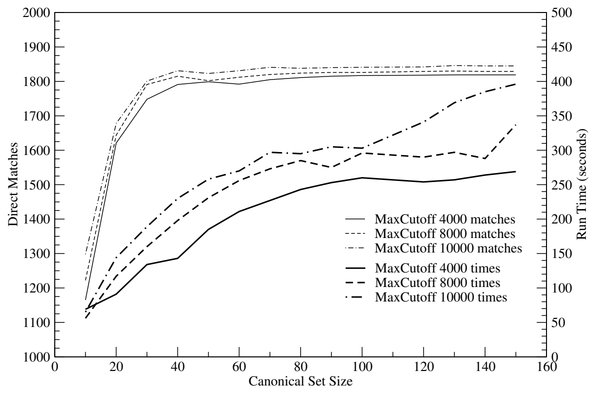

Figure 5: Performance against canonical set size

8 Conclusion

In these experiments we have proposed new measure and weight functions that, as our evaluation has shown, significantly outperform existing similarity functions. The list of measure and weight functions we compared against is not complete, and we hope to add other functions to provide a general framework for thesaurus extraction experimentation. We would also like to expand our evaluation to include direct methods used by others (Lin, 1998a) and using the extracted thesaurus in NLP tasks.

We have also investigated the speed/performance trade-off using frequency cutoffs. This has lead to the proposal of a new approximate comparison algorithm based on *canonical attributes* and a process of coarse- and fine-grained comparisons. This approximation algorithm is dramatically faster than simple pairwise comparison, with only a small performance penalty, which means that complete thesaurus extraction on large corpora is now feasible. Further, the canonical vector parameters allow for control of the speed/performance trade-off. These experiments show that large-scale thesaurus extraction is practical, and although results are not yet comparable with manually-constructed thesauri, may now be accurate enough to be useful for some NLP tasks.

Acknowledgements

We would like to thank Stephen Clark, Caroline Sporleder, Tara Murphy and the anonymous reviewers for their comments on drafts of this paper. This research is supported by Commonwealth and Sydney University Travelling scholarships.

References

L. Douglas Baker and Andrew McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, Australia, 24–28 August.

John R. L. Bernard, editor. 1990. *The Macquarie Encyclopedic Thesaurus*. The Macquarie Library, Sydney, Australia.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 120–126, College Park, MD USA, 20–26 June.

Stephen Clark and David Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 95–102, Pittsburgh, PA USA, 2–7 June.

Carolyn J. Crouch. 1988. Construction of a dynamic thesaurus and its use for associated information retrieval. In *Proceedings of the eleventh international conference on Research and Development in Information Retrieval*, pages 309–320, Grenoble, France, 13–15 June.

James R. Curran and Marc Moens. 2002. Scaling context space. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, Philadelphia, PA USA, 7–12 July. *(to appear)*.

Cristiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. The MIT Press, Cambridge, MA USA.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston, USA.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th international conference on Computational Linguistics*, pages 539–545, Nantes, France, 23–28 July.

Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics and of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 768–774, Montréal, Québec, Canada, 10–14 August.

Dekang Lin. 1998b. An information-theoretic definition of similarity. In *Proceedings of the Fifteen International Conference on Machine Learning*, pages 296–304, Madison, WI USA, 24–27 July.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA USA.

Scott McDonald. 2000. *Environmental determinants of lexical processing effort*. Ph.D. thesis, University of Edinburgh.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust applied morphological generation. In *In Proceedings of the First International Natural Language Generation Conference*, pages 201–208, 12–16 June.

Darren Pearce. 2001. Synonymy in collocation extraction. In *Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations, (NAACL 2001)*, pages 41–46, Pittsburgh, PA USA, 2–7 June.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st annual meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio USA, 22–26 June.

Peter Roget. 1911. *Thesaurus of English words and phrases*. Longmans, Green and Co., London, UK.

Gerda Ruge. 1997. Automatic detection of thesaurus relations for information retrieval applications. In *Foundations of Computer Science: Potential - Theory - Cognition, Lecture Notes in Computer Science*, volume LNCS 1337, pages 499–506. Springer Verlag, Berlin, Germany.

Mark Sanderson and Bruce Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 206–213, Berkeley, CA USA, 15–19 August.

Grady Ward. 1996. *Moby Thesaurus*. Moby Project.