Using similarity measures to extend the LinGO lexicon

Lynne Cahill

Natural Language Technology Group
University of Brighton
Brighton BN2 4GJ, UK
L.Cahill@brighton.ac.uk

Abstract

Deep processing of natural language requires large scale lexical resources that have sufficient coverage at a sufficient level of detail and accuracy (i.e. both recall and precision). Hand-crafted lexicons are extremely labour-intensive to create and maintain, and require continuous updating and extension to retain their level of usability. In this paper we present a technique for extending lexicons using similarity measures that can be extracted from corpora. The technique involves creating lexical entries for unknown words based on entries for words that are known and that are deemed to be distributionally similar. We demonstrate the usefulness of the approach by providing an extended lexicon for the LinGO system using similarity measures extracted from the BNC. We also discuss the advantages and disadvantages of using such lexical extensions in different ways – principally either as part of the main lexicon or as a separate resource used only for "last resort" use.

1. Introduction

NLP applications that require deep semantic analysis or generation from sophisticated semantic representations can rarely get by without large scale lexical resources. Although statistical methods may produce excellent results for certain (especially parsing) tasks without the benefit of such resources, there are other tasks which simply must have lexical resources of this type. This paper presents a method for automatically extending existing lexical resources for such applications.

The LinGO system can be used to either parse or generate. In parsing mode, it takes sentences of English and produces a representation of the semantics in the form of Minimal Recursion Semantics (Copestake, 2005). In generation mode, it takes an MRS and produces an English sentence. The system comes with a wide coverage grammar (known as the English Resource Grammar, or ERG) consisting of HPSG grammar rules and a lexicon of around 22,000 words.

In order to extend the lexicon, syntactic as well as semantic information is required for the words to be included. This information can be found in a number of different types of resource, such as electronic dictionaries, WordNet (Felbaum, 1998) etc.. The problem with many of these resources is that they do not provide the precise information needed, or do not provide it in a form that is easily accessible. WordNet, for example, gives us semantic information, but lacks the syntactic information needed.

What we do have available to us is software, developed at the University of Sussex (Weeds and Weir, 2005), which can be used to extract, from any corpus, information about the distributional similarity of the words that occur in the corpus. If we can harness this resource to generate lexical entries based on existing entries for words with a high degree of similarity, we have the potential to extend lexicons like the LinGO one almost infinitely.

After reviewing alternative approaches to lexical extension, we first look at the LinGO system and the distributional similarity software. We then present the method used to generate new lexical entries. Next, we present an evaluation of the usefulness of the technique for parsing within the LinGO system. Finally, we discuss issues of how, exactly, such automatically generated resources should be used and the implications for precision and recall in the NLP tasks that they are used for.

2. Previous work

There have been a number of different approaches suggested to overcome the problem of lexical gaps. These fit broadly into two different categories: (1) approaches that dynamically try to analyse unrecognised lexical items and (2) approaches that try to extend the lexicon to avoid gaps while processing. This section looks only at those approaches that are geared towards the specific type of grammar/lexicon in which we are interested.

(Baldwin et al., 2004) describes an approach to extending the lexicon in the ERG manually, after ”road-testing” it on the BNC. This approach is very much in the second category above. It involves much more manual effort, and understanding of the inner workings of LinGO (it makes use of additional lexical rules, for example, as well as just adding entries to the lexicon). It provides tools that enable grammar/lexicon developers to extend the lexicon, but it as crucially a manual rather than automatic task.

(Zhang and Kordoni, 2006) fits into the first category above, dynamically creating lexical entries for unknown words during parsing. What makes their approach distinct from ours is primarily that they are concerned with assigning lexical types, rather than matching individual lexical entries. Depending on the structure of your lexicon and the size and nature of the set of lexical types used, this might not be a very significant difference, but our approach allows us to generate entries based on a single entry rather than a whole set.

Other approaches that don't quite fit into either category include, (Barg and Walther, 1998) and (Fouvry, 2003), who use the notion of underspecified lexical entries to allow for partial matching of unknown words. These approaches tend to involve dynamically generating lexical entries which are then added permanently to the lexicon.

3. LinGO

LinGO (Copestake and Flickinger, 2000) is a combined parser/generator which comes with a range of resources, including the English Resource Grammar (ERG)[1]. This grammar consists of HPSG grammar rules for English and a lexicon of around 22,000 words. Although the coverage is reasonable when it comes to frequent words, a lexicon of this size will never be adequate for a wide range of different domain specific texts, nor for open domain texts. Thus, anyone wishing to use LinGO will typically need to extend the lexicon to include the words required[2].

The LinGO parser takes sentences of English and produces representations in Minimal Recursion Semantics (Copestake, 2005). These representations can be given to the LinGO generator to produce sentences of English. If a word in a sentence is not present in the LinGO lexicon, the system will fail to produce a parse. Similarly for generation, if there is no lexical entry with the required semantic form, then no sentence will be generated. Thus, the robustness of the system depends on adequate lexical resources.

The lexical entries are relatively simple, providing a part of speech classification, the stem, a more detailed syntactic classification and a semantic representation. In addition, each lexical entry states whether the stem begins with a vowel or a consonant (necessary for combining rules such as a/an alternation).

As is well documented, lexical extension can be a very labour-intensive task, and demands appropriate resources, such as dictionaries, to be available. We therefore propose a method of extending the LinGO lexicon that makes use of existing technology, and which can be applied to information extracted from any corpus.

The documentation that comes with the LinGO system (and specifically the English Resource Grammar) stresses the importance of lexical extension, explaining that the lexicon has been extended manually and "the lexicon now contains all verb subcat entries for the 2000 most frequent verb stems in the British National Corpus (BNC). This should enable some interesting experimentation in automated lexical acquisition, since there are fewer lexical types that need to be hypothesized for non-verbs."[3]. This is interesting from our point of view as it suggests exactly the kind of experiment we are undertaking.

4. Distributional similarity

(Weeds and Weir, 2005) provides a flexible framework for extracting measures of distributional similarity from corpora. The framework allows parametrised analysis of distributional similarities using a variety of different possible context-types. The similarity measures we make use of here are those based on grammatical function, implemented initially to contruct (or extend) thesaurus-type resources. This is most appropriate for our purposes, as we ideally want words which are at least likely to be semantically as well as syntactically similar. The output of this process is a file of triples: the target word, the similar word and the similarity measure. For example, a sample run applied to the BNC and restricted to the most frequent 2000 nouns results in a file with triples as in figure 1.

Note that the first entry for any target word lists it as having a similarity value of 1 with itself. It should also be noted that, although it is usually the case that words that exhibit very high distributional similarity are also semantically similar, this is not necessarily the case. The actual values (for non-identical words) are rarely much above 0.4, so we are typically looking at similarity measures between 0 and 0.4.

5. Generating lexical entries

We can now use the output of this process to generate additional lexical entries. First we must decide how we wish to integrate the additional lexical entries. The most simple approach involves adding the newly generated entries to the existing lexicon, so that they are used in the same way as other entries. At least two other approaches are possible. The first of these is to have two separate lexicons, one of which is queried first, with the second used only as a back-up, if no entry is found in the first. The second is to generate the entries dynamically, only as they are required. To date we have only tested using the first approach, but this is risky, as we would rather give preferential status to the lexical entries in which we have most confidence (i.e. the hand crafted entries). We will discuss this issue further in section 7.

The next decision we need to make is how to choose the new words and the existing lexical entries to base them on. Given the nature of the distributional similarity software, the number of words that are given similarity measures for each head word can be very large. The first word in the example list above, for example, "importance" has a total of 1996 similarity values defined, that is one for almost all of the 2000 nouns included in this particular set. The similarity measure for the last of these ("globe") is 0.0022. This does not, of course, mean that there is the potential for 1996 additional entries for our head word. This is because, firstly, many of these 1996 words will not have entries in our existing lexicon either. Secondly, many of the entries are so similar that the new entries that we generate based on them are identical. In view of the fact that there will be some measure (potentially very small) of similarity between virtually any word of the same major word class, we do not wish to generate entries for every possible word pair. In our experiments we have tested adding only a single entry, adding five entries and adding twenty entries. This does not necessarily mean that we add entries for the top 1, 5 and 20 similarity pairs, but we add the first 1, 5 and 20 distinct entries that are possible, given the ordered list of pairs.

Another question that we need to address at this point is the question of word class. The similarity software deals with a single word class at a time, so we have the choice of using the output in batches distinguished by their wordclass, or collating the output from the similarity measures and us-

[1]We are using the release dated 05/06/2005.

[2]A significant proportion of the LinGO lexicon entries are for multi-word expressions, rather than inidivdual words. Although this is an interesting area, it is beyond the scope of the present project, so our discussions will be confined to lexical entries for single words.

[3]See lingo.stanford.edu – README file in the ERG grammar component.

```
"importance"    "importance"    1.0
"importance"    "significance"  0.41190367926325505
"importance"    "influence"     0.35182572460391287
"importance"    "extent"        0.3379548895057337
"importance"    "need"          0.31725514633546115
"importance"    "strength"      0.31093171330002195
"importance"    "value"         0.30984416598432163
"importance"    "status"        0.3000272835800818
"importance"    "role"          0.29884422333029836
"importance"    "impact"        0.2980094466854172
```

Figure 1: A sample of output similarity measures

ing it as a single resource. We have opted to do the first of these, mainly because of the relatively sparse information that we actually have access to at the end of the similarity measuring process. It would be very difficult to make principled decisions regarding the relative appropriateness of generating lexical entries for nouns based on the similarity of verbs with the same stem.
As a simple example, let us consider the word “acclaim”. If we consider this as just a noun, we get similar words including “applause”, “reward”, “remuneration” and “patronage”. If we consider “acclaim” as a verb, only “reward” of this set could be a potential basis for a new lexical entry. However, if we do not separate the process by word class, there would be the potential for confusion, with a word like “patronage” possibly getting a lexical entry for it as a verb, based on the verb entry for “acclaim”.

Running the program

The lexical extension program is written in AWK and makes extensive use of arrays, reading the content of one file into an array for use while processing the other file. The program takes three inputs: the file with the existing lexicon (which is read into an array), the file with the similarity measures (which is the main input to the AWK script) and a value indicating the word class (currently only either verb or noun). The output is a separate file which includes the lexicon extension with just the new entries. This lexicon file can simply be slotted in to your LinGO implementation and used as normal.

Reading in the old lexicon

The first thing the program does is to read in the existing lexicon, constructing an array with all the lexical information indexed on the stem forms[4]. The array consists of five pieces of information: the head, the part of speech classification, the syntactic classification, the semantics and whether the stem begins with a vowel or a consonant. The head is the head of the entry in the initial lexicon. This is usually nearly the same as the stem, but often with subscript markers to distinguish more than one entry with the same stem. The part of speech is a LinGO classification and distinguishes between, for example, proper nouns and common nouns, mass and count etc. and for verbs it specifies the basic subcategorisation classification. The second piece of syntactic information defines the relationship between the syntax and semantics, and consists of a label such as LKEYS.KEYREL.PRED (which indicates that this word contributes the predicate). Finally, the semantic information takes the form of a relation/predicate name. Typically, this is based on the stem form, so that a noun like “ditch” has the semantics “_ditch_n_rel”, indicating that its semantics behaves like a noun relation (n_rel). The simplicity of the majority of these semantic relations is one of the key points that makes the automatic extension of the lexicon possible.
The array indexes on the head, not on the stem, because more than one entry may have the same stem, whereas the subscripts differentiate the heads. As an example, there are two lexical entries in the existing LinGO lexicon for the word “follow”. The first of these is:

```
follow_v1 := v_np*_trans_le &
 [ STEM < "follow" >,
   SYNSEM [ LKEYS.KEYREL.PRED
              "_follow_v_1_rel",
   PHON.ONSET con ] ].
```

The array elements from this entry are defined as follows:

lexentry[follow_v1,1] = follow
lexentry[follow_v1,2] = v_np*_trans_le
lexentry[follow_v1,3] = LKEYS.KEYREL.PRED
lexentry[follow_v1,4] = _follow_v_1_rel
lexentry[follow_v1,5] = con

where the numbers 1-5 indicate which piece of lexical information we are dealing with. 1 is the stem, 2 is the part of speech, 3 is the syntactic relation, 4 is the semantic relation and 5 is the phonological onset of the stem.
In addition to this, we create another array which simply lists the stems that are included in the lexicon, so that we can very easily discover whether or not a word appears in the lexicon already.

Determining the need for new entries

The program runs on the file of distributional similarities, rather than on the existing lexicon. In this file, a record consists of three fields: the target word, the word to which it is similar and the measure of how similar it is. For each

[4]The lexicon in the distribution of the ERG is available as both a fully formatted lexicon file and as a lexical database, with comma separated fields representing the information. We use the latter to create the array comprising the information inthe lexicon.

record we first check whether the word appears in the lexicon. If it does, we do nothing more. If it does not, we then check whether the other word does appear in the lexicon. If it does not, there is nothing more we can do. If it does, we create a new lexical entry (or entries) for the target word based on the entry (or entries) for the other word.

There are a number of ways in which this procedure could be varied. For a start, the initial decision not to continue if there is an existing lexical entry for a word may not be correct. In the case of nouns, it is unlikely that this will be problematic. However, when it comes to verbs, where several different subcategorisation options may apply to a single stem, the situation is very different. The fact that a single lexical entry is generated for a stem does not necessarily mean that occurrences of that stem in sentences will be correctly parsed (or even parsed at all). On the other hand, we do not want to generate spurious additional entries where they are not needed. We have, as mentioned above, conducted three separate experiments including one, five and twenty new entries.

The second part of this question relates to how many of the existing entries for a stem we may want to use as templates for our new entry/ies. For example, we have two existing lexical entries for the verb "follow":

```
follow_v1 := v_np*_trans_le &
 [ STEM < "follow" >,
   SYNSEM [ LKEYS.KEYREL.PRED
              "_follow_v_1_rel",
   PHON.ONSET con ] ].

follow_v2 := v_expl_it_subj_cp_le &
 [ STEM < "follow" >,
   SYNSEM [ LKEYS.KEYREL.PRED
            "_follow_v_expl_rel",
   PHON.ONSET con ] ].
```

The first of these represents the standard transitive use of the verb as in a sentence like "The cat follows the dog". The second represents the it-cleft use as in "It follows that ...". If we have a transitive verb, such as "chase", which has no entry in our lexicon, but which we know to be similar to "follow", we need to determine which of these two entries would be appropriate. The information we have from the similarity measures does not involve enough detail to allow us to make principled decisions according to the exact kind of syntactic distribution that the words share, so we are not in a position to decide that the transitive use is the appropriate one. This leads to a difficult decision between using both entries, and thereby generating an incorrect one that would allow the sentence "It chases that ...", or using only one, but potentially excluding useful entries. We will discuss this further in section 7.

There is clearly a difference in the relative importance of different bits of lexical information for parsing and generation. It is more likely that a single lexical entry will help the parsing process than the generation process, as the generation of a sentence from a MRS requires exact combinations of semantic predicates/relations. When parsing, on the other hand, it is often necessary only to determine the approximate syntactic information in order to complete a parse. We anticipate, therefore, that when we undertake full testing for both parsing and generation, the method will prove more useful for parsing than generation, and more work will be require to fully extend the lexicon for generation tasks. To date we have only experimented with parsing.

Creating the new entry

In the final stage of the process, we construct in our array an additional entry that includes a new lexical head, constructed from the stem form, with the additional marker indicating its word class (i.e. _v or _n) and the number of the entry. As discussed above, in the first instance, this will always be 1, as we are only generating single entries for each unknown word, but it is still necessary to include the number in case future extensions add further entries. So, for the example above, we will include in our array the information that the stem of the new entry is "chase". Next, we will add the part-of-speech information, which we take from the existing entry for "follow", which is "v_np*_trans_le". The syntactic information that forms part of the "synsem" is again taken from the existing entry, i.e. "LKEYS.KEYREL.PRED". The semantic representation is constructed partly from the existing entry and partly from the stem form. The semantics for this entry for "follow" is "_follow_v_1_rel" which we take and adapt by simply swapping the stem of "chase" with the stem of "follow". We approximate the phonological information from the orthographic form, which will inevitably lead to occasional errors (the words *usable* and *unusable*, for example, require different specifications for their initial sound), but these are rare.

We therefore define the new array elements as follows:

lexentry[chase_v1,1] = chase
lexentry[chase_v1,2] = v_np*_trans_le
lexentry[chase_v1,3] = LKEYS.KEYREL.PRED
lexentry[chase_v1,4] = _chase_v_1_rel
lexentry[chase_v1,5] = con

This is all we need to do for each individual entry, as the entire lexicon is going to be output from the array.

Creating the new extended lexicon

When our program has finished running through the similarity measure file, it has produced a very large array with all of the lexical information in it. The final task is to output the information in LinGO lexical entry format. For this, we simply slot the values defined in the array into the template for each lexical entry.

We output the new entries into a separate lexicon file, which is loaded after the main lexicon. This has already been implemented as part of the COGENT project, with a set of hand-crafted lexicon entries for the domain of medical texts (patient information leaflets).

6. Evaluation

There are (at least) two types of evaluation required for this exercise: (1) Are the additional lexical entries correct/appropriate? and (2) Do the additional lexical entries improve the performance of LinGO? The first of these we have checked by manually examining the new entries produced. For the experiment where we added a single entry per unknown word, this involved 445 new entries. We checked a random sample of around 50 of these, and no major issues emerged.

In order to fully evaluate the usefulness of the method it is necessary to compare the performance of the LinGO parser (or generator) before and after the lexical enhancement. This is, however, a lengthy and labour intensive task. We have run a series of experiments aimed at determining the level of performance of the LinGO parser with the original lexicon and with various lexicons extended by means of the software described here.

We wanted to test the parser on a small, general and relevant corpus. As the similarity measures had been determined from the BNC, we decided to take a small subset of this corpus to run the parser on. We chose a subfile of the BNC at random (the AD0 section). Having removed all of the markup, we were left with 2018 sentences, with an average sentence length of 18.9 words. This subcorpus has a total of 38,125 words and 3773 distinct tokens.

The base lexicon we used had been extended for use in the COGENT project with the addition of a total of 424 entries relating to the medical field. We ran four experiments: with the base lexicon; with the base lexicon extended with a single entry for each unknown word (LinGO+1); with the base lexicon entended with the first five entries for each unknown word (LinGO+5); and with the base lexicon extended with the first twenty entries for each unknown word (LinGO+20). The results for full parses can be seen in Figures 2 and 3.

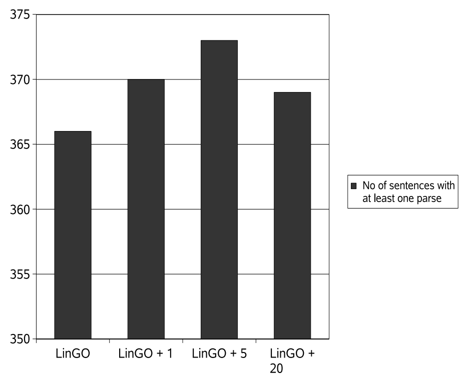

Figure 2: Number of sentences with at least one full parse

The most obvious thing to note is that the system does not manage to find full parses for a very large percentage of the sentences. In the base case, 366 of the 2018 sentences received full parses, just over 18%. However, it should be stressed that the system had not been tuned for this corpus, and the requirement for full deep parses mean that this is a very difficult task. The next thing to note is that our extended lexicons do not make a very large difference. Again, this is not something that we should be surprised about. The actual numbers of new entries in this experiment is very small, particularly when we consider Zipf's law, so adding 445 entries for nouns that we assume not to be the most frequent (or they would already have been in the 22,000 word lexicon) would not be predicted to make a very big difference to the number of sentences parsed.

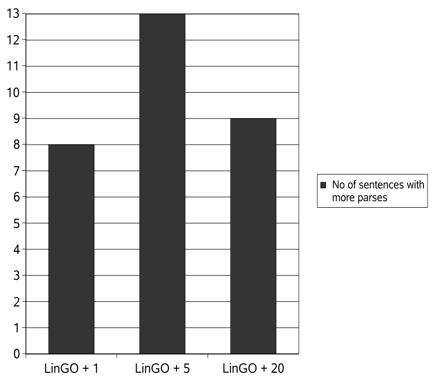

Figure 3: Number of sentences with more parses than the base

Another thing that stands out from these results is that extending the lexicon by twenty entries for each new word actually causes the performance to decline, compared to the other extended lexicons. This is, we believe, because of other features of the LinGO system, specifically the inbuilt cut-off for the number of edges, which leads to the system failing when too many different edges are being considered. This is therefore an issue that requires more detailed analysis of the LinGO parsing algorithm.

So we see from this table that the number of full parses increased by four with the LinGO+1 lexicon and by a further three with the LinGO+5 lexicon. This is around a one percent increase in each case. However, if we look more closely at the parser's performance we see evidence that the lexical extensions may be contributing more.

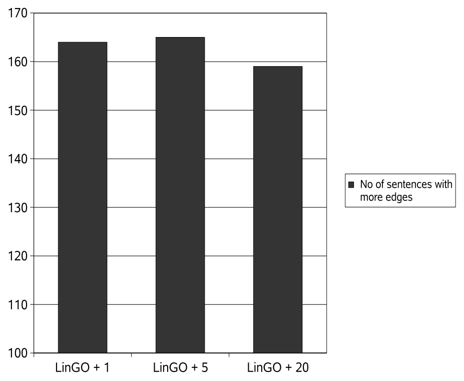

Figure 4: Number of sentences with more edges than the base

Figure 4 shows the number of sentences in which the parser produced more edges than in the base system, even though it might not have resulted in a complete parse. While it is clearly not necessarily a good thing to generate large numbers of edges, it shows that the parser is able to progress further. It will take much longer to examine the full and partial parses to establish whether the edges generated are

correct and why full parses are not resulting.
Figure 5 shows how the number of words that are unrecognised is reduced with the extended lexicons.

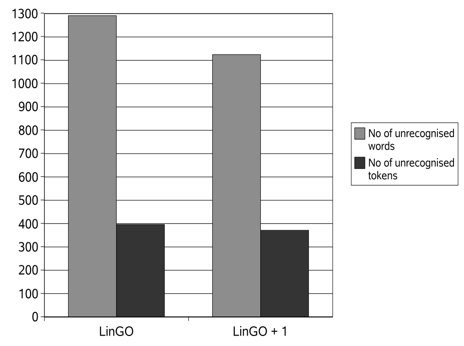

Figure 5: Number of unrecognised words

The number of distinct tokens unrecognised in the base lexicon is 396. This is reduced by 25 in the extended lexicons[5]. These 25 words represent 0.4% of the total words in the corpus, emphasising that the words we are dealing with here are not of the highest frequency, but not insignificant either.

7. Conclusions and discussion

This paper has described a small experiment to prove that it is possible to usefully extend a lexicon for a deep parsing system such as LinGO by using distributional similarity measures. The results show that, while there is much room for improvement in many directions, the approach appears to provide a potentially very useful solution to the problem of robustness in deep processing systems.
One thing that the figures above make plain is the importance of lexical coverage in the deep parsing task. The number of unrecognised words in the base system is 396, which amounts to 10.5% of the total number of distinct tokens in the corpus. The leads to a full parsing failure rate of 82%.

Confidence

In order to give the greatest robustness to the LinGO system, we would need to allow the use of entries in which we do not necessarily have such great confidence, if that is the only option. Currently the extensions to the lexicon are loaded into the parser with equal status to the original lexical entries. One of the features of the LinGO parser is that it often generates a number of parses as well as a large number of partial parses. Generating spurious parses because of unreliable entries is clearly, therefore, undesirable. The crucial task is to get the right balance between precision and recall. Accessing the less reliable lexical entries only when no parse can otherwise be found would clearly help in reducing the number of spurious parses.

[5]The number is the same for all of the extended lexicons as they do not differ in the number of distinct tokens represented, only in how many different entries there are for that token.

We are therefore developing an approach which allows two-stage lexical lookup, where the automaticaly generated entries are only used where no parse is found initially. We have used a similar approach in earlier parsing systems, where complete syntactic parses were required but semantic information was only needed for words relevant to the domain. The difference here is that we need the lexical entries in which we have less confidence to provide us with semantic as well as syntactic information.
Another possible future development is to allow the lexical extension software to have access to much more information from the distributional similarity software, or even call it interactively. This would mean that the two processes could run together on any new corpus, and the parser would be dynamically tuned to the corpus on the basis of the distributional similarity of the words in that corpus. It would also give an excellent opportunity to directly test the performance of different approaches to distributional similarity for this particular task.

Parsing vs generation

The experiments we have undertaken so far have only looked at improving the parsing in LinGO. We have mentioned at various points above how the issues may be different for generation. We shall now look further at this issue.
When parsing, a system can often perform adequately even without full (syntactic and semantic) lexical information for every word in the sentence. As mentioned above, there are a range of systems which make use of very large scale shallow lexical resources (e.g. syntactic class information only), or even POS tagging software to provide the parser with the minimum information required. Although this is not the case with LinGO, it is nevertheless still true to say that the parser can be expected to perform reasonably, even if the semantic information included in additional lexical entries is incomplete or even slightly inaccurate. This is not the case with genreation.
Consider the generation task. In order to generate a sentence from an underlying semantic representation it is essential to have some means of representing every piece of information in the semantic representation. As language users, we very commonly "understand" sentences uttered to us, even if we miss one or two of the words. However, we do not typically find ourselves in the position of uttering incomplete sentences because we can't think of the word we need.
The way in which we go about finding potential lexical extensions for the generation task would, in addition, be more complex. We would need to look at the MRSs from which we wished to generate and determine any semantic components that had no equivalent in the lexicon. But we would then need to find equivalent entries based on the relationship between the semantic component and the surface realisation. This is far less straightforward than the parsing and examination of output that we have undertaken in the current experiment.
The first step for us in this direction is to carefully handcraft a set of MRSs which would require entries that we have generated automatically and see if they generate the sentences we want and expect.

Other parsing systems

We have presented the lexical extension technique as tested on the LinGO ERG grammar and lexicon. We believe, however, that the technique could equally be applied to different grammars (e.g. for different languages) and different systems. The novel part of our approach is to directly make use of distributional similarity measures to determine relations between unknown words in a corpus and known words in that corpus to produce lexical entries modelled on the known word. The aspects of the programme that make it specific to LinGO ERG are really the insubstantial (cosmetic) aspects of outputting formatted lexical entries. The only area where there must be commonality is in the type of lexical information that is specified. It does not matter to the lexical extension software what, for example, the syntactic or semantic labels are, only that they are defined. The initial phase of reading in exisiting lexical information is, similarly, not likely to be difficult to adapt for other lexical database formats.

Therefore, although this is not a general purpose tool that could be run "off-the-shelf" on other systems, the key elements of the technique could be applied to other systems and grammars.

Acknowledgements

This work was supported by EPSRC grant GR/S24480/01. I would like to thank my colleagues on the COGENT project at the Universitites of Brighton and Sussex for their support. I would also like to thank the anonymous reviewers for helpful comments on the abstract, and an earlier version of the paper.

8. References

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the english resource grammar over the british national corpus. In *Proceedings of the fourth International Conference on Language Resources and Evaluation*, pages 2047–50, Lisbon, Portugal.

Patra Barg and Markus Walther. 1998. Processing unknown words in hpsg. In *Proceedings of the 36th International Conference of the ACL and the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad coverage english grammar using hpsg. In *Proceedings of the Second International Conference on Language Resources and Evaluation*.

Ann Copestake. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.

Christiane Felbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge.

Frederik Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *Companion to the 10th EACL*, pages 87–90, Budapest, Hungary.

Julie Weeds and David Weir. 2005. Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31:4:439–476.

Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open texts processing. In *Proceedings of the fifth international conference on Language resources and evaluation*, pages 275–280, Genoa, Italy.