Intermediate Parsing for Anaphora Resolution? Implementing the Lappin and Leass non-coreference filters

Judita Preiss and Ted Briscoe*
Computer Laboratory, University of Cambridge
`Judita.Preiss@cl.cam.ac.uk`, `Ted.Briscoe@cl.cam.ac.uk`

Abstract

We show that the Lappin and Leass (1994) anaphora resolution algorithm, originally implemented to use McCord's Slot Grammar, can be equivalently expressed in terms of grammatical relations (GRs) (Carroll et al., 1998). We use this GR version of Lappin and Leass' algorithm to investigate the effect on performance when the parser is changed. We find that the performance of the anaphora resolution algorithm does not appear to be greatly affected. The results from the GR version of the Lappin and Leass anaphora resolution algorithm are also briefly compared to the results of the Kennedy and Boguraev (1996) anaphora resolution algorithm.

1 Introduction

We question the need to use a full parser, such as McCord's Slot Grammar, in the Lappin and Leass (1994) anaphora resolution algorithm. The goal of our work is to show that this algorithm can be equivalently implemented in terms of Carroll et al.'s (1998) grammatical relations (GRs). By a full parser we understand a system which uses subcategorization information, and it is not necessary to construct such a full parse for a system which outputs GRs. The new implementation enables us to investigate performance variation in the Lappin and Leass algorithm when the parser is changed for any other GR producing parser. This work is motivated by our previous work (Preiss, 2002), which showed that the performance of the anaphora resolution algorithm did not change greatly if the full parser it uses was varied, so long as the grammatical roles could be extracted comparably accurately from the parsers.

As noted by Kennedy and Boguraev (1996) and still true today, it is a fact that state-of-the-art full parsers are not yet robust enough for an application like anaphora resolution. As we have observed in our own work, if an anaphora resolution experiment is restricted to blocks of sentences for which a full parse can be successfully generated, the anaphora resolution results turn out 'too good'. The unrealistic bias reflects the grammatical simplicity. On the other hand, it is hard to design anaphora resolution experiments if the parser fails on some sentences, since the correct antecedent might be hidden in a sentence without a parse. Since parsers which return labelled headword dependent-word links (intermediate parsers) are not obliged to return a full parse, they generally do not fail completely on sentences, but instead return fragments of a parse. This robustness makes it attractive to investigate whether intermediate parsers can be applied in an anaphora resolution algorithm.

Kennedy and Boguraev managed to eliminate the need for a full parse, however their algorithm was a substantial modification of the

*This work was supported by UK EPSRC project GR/N36462/93 'Robust Accurate Statistical Parsing (RASP)'.

Lappin and Leass algorithm. It is not clear whether this was necessary. It was pointed out by Barbu and Mitkov (2001) in their comparison of anaphora resolution systems, that performance can only be compared if the systems are evaluated on the same corpus and share as many tools as possible. The GR version allows us to use a single implementation of the Lappin and Leass algorithm, and just to change the set of GRs associated with the evaluation corpus. This allows us to directly compare the performance difference and analyze errors. We also present the results of the Kennedy and Boguraev anaphora resolution algorithm on our corpus.

In Section 2 we describe the Lappin and Leass anaphora resolution algorithm and our implementation which only uses GRs. A comparison of the two intermediate parsers that we employ can be found in Section 3. The results of the anaphora resolution algorithm using the two parsers are presented in Section 4. This section also contains the results for the Kennedy and Boguraev algorithm, and an analysis of the results. In Section 5 we draw our conclusions.

2 Lappin and Leass Algorithm

2.1 Algorithm Description

For this experiment we choose to re-implement a non-probabilistic algorithm due to Lappin and Leass (1994), since this anaphora resolution algorithm mainly makes use of grammatical role information, and therefore only minor modification would be required for an intermediate parser to be used instead of the full parser they assume.

For each pronoun, this algorithm uses syntactic criteria to rule out noun phrases that cannot possibly corefer with it. An antecedent is then chosen according to a ranking based on salience weights.

For all types of pronoun, noun phrases are ruled out if they have incompatible agreement features. Pronouns are split into two classes: lexical (reflexives and reciprocals) and non-lexical anaphors. There are syntactic filters for the two types of anaphors.

Factor	Weight
Sentence recency	100
Subject emphasis	80
Existential emphasis	70
Accusative emphasis	50
Indirect object/oblique	40
Head noun emphasis	80
Non-adverbial emphasis	50

Table 1: Salience weights

Candidates which remain after filtering are ranked according to their salience. A salience value is computed to be a weighted sum of the relevant feature weights (summarized in Table 1). If we consider the sentence *John walks*, the salience of John will be:

$$\begin{aligned}\text{sal(John)} &= w_{\text{sent}} + w_{\text{subj}} + w_{\text{head}} + w_{\text{non-adv}} \\ &= 100 + 80 + 80 + 50 \\ &= 310\end{aligned}$$

The weights are scaled by a factor of $\left(\frac{1}{2}\right)^s$ where s is the distance (number of sentences) of the candidate from the pronoun.

The candidate with the highest salience is proposed as the antecedent.

2.2 Using Grammatical Relations

The Lappin and Leass anaphora resolution algorithm exploits two types of information: information which is directly obtained from GRs (for example, being a subject or direct object), and information for which extraction is not so simple (for example, whether the pronoun is in the argument or the adjunct domain of a noun phrase). The original implementation of Lappin and Leass' algorithm relied mainly on a clausal (head–argument) representation of McCord's Slot Grammar (Lappin and McCord, 1990). Kennedy and Boguraev (1996) modified the Lappin and Leass algorithm to use flat morpho–syntactic information (a shallow parser), this being more flexible for new domains.

Our implementation of Lappin and Leass uses (only) GRs conforming to the specifica-

Sentence: Mary gave her a book.

Grammatical Relations:

```
(ncsubj gave Mary _)
(dobj gave her _)
(obj2 gave book)
(detmod _ book a)
```

Figure 1: Briscoe and Carroll output for *Mary gave her a book*

tion of Carroll et al. (1998). An example for the sentence *Mary gave her a book* can be seen in Figure 1. The underscores in the GRs above indicate the GR's type. For example, *detmod* has type *poss* for pronominal determiners, e.g. `(detmod poss cat his)`. Because it is possible to recover constituency from a set of GRs (Hays, 1964), (Gaifman, 1965), all information necessary for the anaphora resolution algorithm is present in the GRs.

Information concerning the grammatical role of each noun phrase in the sentence can easily be recovered from the GRs. However, it is not immediately clear that *Mary* and *her* cannot corefer. This is decided by one of a number of syntactic constraints in the Lappin and Leass algorithm, which rule out certain intrasentential noun phrases from becoming candidate antecedents.

As the success of the algorithm is mainly dependent on the early filtering of candidates, a substantial part of our work focused on obtaining this information from the GRs. For example, the argument domain filter which rules out coreference in the above example can be encoded by:

```
(arg - X N -)
(arg - X P -)
```

where arg $\in \{ncsubj, dobj, iobj, obj2, xcomp^1\}$. In this case, - means that the value of the type is unimportant.

An adjunct domain filter, ruling out coreference in *She sat near her*, can be expressed as:

```
(arg   - X N -)
(ncmod Prep X P)
```

where arg $\in \{ncsubj, dobj, iobj, obj2, xcomp^2\}$.

In this way we have encoded all the Lappin and Leass syntactic constraints. They were found to depend on the object relations (*ncsubj*, *dobj*, *obj2*, *iobj*), the complement relations (*xcomp*, *ccomp*, and *clausal*), and the non-clausal modifier *ncmod* relation. We also use the GRs to extract the salience factor information.

3 Intermediate Parsing

3.1 Grammatical Relations

We use the GRs obtained from the Briscoe and Carroll (2002) parser and the Buchholz (2002) GR finder for the parser components of our implementation of the Lappin and Leass anaphora resolution algorithm. Both these parsers are capable of producing GRs according to the Carroll et al. (1998) specification. We will now describe the two parsers, and present their performance on this corpus.

3.2 Briscoe and Carroll (2002)[3]

Grammar: unification-based (manually created) grammar of part of speech and punctuation labels.

Parsing algorithm: LR parser.

Tagger: Acquilex HMM tagger (using the CLAWS-II labels) (Elworthy, 1994).

Training corpus: Susanne corpus is used for development (Sampson, 1995).

3.3 Buchholz (2002)

Tagger: Memory-based tagger due to Daelemans et al. (1996).

Chunker: Memory-based chunker due to Veenstra and van den Bosch (2000).

Shallow Parser: Memory-based shallow parser (Daelemans, 1996), (Buchholz et al., 1999).

[1] *xcomp* is only relevant when H is a form of *be*.

[2] Similarly, *xcomp* is only relevant when H is a form of *be*.

[3] Available from `http://www.cogs.susx.ac.uk/lab/nlp/rasp/`

GR	# occ	BC		BU	
		Precision	Recall	Precision	Recall
ccomp	81	20.45	11.11	65.00	64.20
detmod	1124	91.15	89.77	92.41	90.93
dobj	409	85.83	78.48	88.42	76.53
iobj	158	31.93	67.09	57.75	51.90
ncmod	2403	69.45	57.72	66.86	51.64
ncsubj	1038	81.99	82.47	85.83	72.93
obj2	19	27.45	73.68	46.15	31.58
xcomp	322	73.99	62.73	78.00	72.67
μ_W	–	75.73	70.29	77.45	66.74

Table 2: GR Precisions and Recalls

Training Corpus: Sections 10–19 of the Wall Street Journal of Penn Treebank II (Marcus et al., 1993).

3.4 Relative Performance

As part of the development of their parser, Carroll et al. manually annotated 500 sentences with their GRs.[4] The sentences were selected at random from the Susanne corpus subject to the constraint that they are within the coverage of the Briscoe and Carroll parser.[5] The performance results of the Briscoe and Carroll parser (BC) and the Buchholz GR finder (BU) for the relevant GRs can be found in Table 2. In this table we also present the (weighted) mean performance μ_W.

We split the 500 sentence corpus into ten 50 sentence segments and computed precision/recall on each segment. The resulting $F_{\beta=1}$ measures were compared using the one-tailed t-test. The GRs produced by the Buchholz GR finder were found to be significantly more accurate than those produced by the Briscoe and Carroll parser (with a confidence of 99.5%).

4 Results

For this experiment, we use a modified version of an anaphorically resolved 2400 sentence initial segment of the BNC (Leech, 1992), (Preiss, 2002). The modification consisted of removing certain tokenizations from the BNC: for example, *out<blank>of* becomes *out of*. To avoid any unwanted tokenization, punctuation was removed from inside of words (e.g. *Binya'a* has become *Binyaa*). We split the 2400 sentence corpus into five texts which contain roughly the same number of pronouns.[6] The number of sentences and pronouns in each of the five texts is presented in Table 3. Pleonastic pronouns were manually labelled as such and were thus removed from consideration by the algorithm (as both anaphors and candidates).

Text	# sents	# prons
1	754	135
2	785	118
3	318	154
4	268	135
5	271	137

Table 3: Corpus Information

In Table 4, we present the results (precision only as all pronouns are always attempted) obtained from our implementation of the Lappin and Leass algorithm using the Briscoe and Carroll parser, and using Buchholz' GR finder. The mean μ of the two 'algorithms' is identical and a one-tailed t-test does not find any significant difference between the two.

[4]Available from `http://www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html`

[5]The Briscoe and Carroll parser has about a 74% coverage of the Susanne corpus.

[6]The texts were created to contain as equal a number of pronouns as possible while not splitting the corpus across any segment boundaries.

Text	# prons	Lappin and Leass		Kennedy and Boguraev	
		BC	BU	BC	BU
1	135	60%	63%	67%	69%
2	118	51%	53%	59%	56%
3	154	70%	70%	73%	71%
4	135	67%	65%	64%	59%
5	137	55%	53%	64%	63%
μ	136	61%	61%	65%	64%

Table 4: Performance Results

In this table, we also present the results for our implementation of the Kennedy and Boguraev algorithm. Although the coreference filters in this case only rely on a shallow parser and thus are not as accurate, the algorithm yields better performance than the Lappin and Leass algorithm. The increase in performance is probably due to the introduction of extra salience factors, and warrants futher investigation.

4.1 Analysis of Results

There is a total of 679 pronouns to be resolved by the Lappin and Leass algorithm, out of which 218 are misclassified by both versions of the algorithm. There are only 95 pronouns which are misclassified by just one of the two versions. In about 40% of the 218 cases, both versions choose the same wrong antecedent which suggests inherent deficiencies of the anaphora resolution algorithm. By inspection, the difference in the remaining 60% of cases was usually due to:

- (At least) one of the sets of GRs being incorrect.
- In the case where the GRs were correct, it was sometimes not possible to choose the correct antecedent according to the anaphora resolution algorithm, due to an inherent deficiency in it.

Interestingly, it is possible for both versions of the algorithm to choose the same antecedent, but for one version this antecedent is correct whereas this is not true for the other. This may occur when the chosen antecedent is a pronoun. For example, in the segment *Mary*... She_1... She_2..., She_2 will resolve to She_1, but She_2 may still be scored wrong if She_1 has not been previously correctly resolved. This evaluation method may lead to chaining errors and may cause the precision of the system to appear lower.

Our results from the Lappin and Leass anaphora resolution algorithm are lower than those originally obtained by Lappin and Leass (85%). However, the original evaluation was carried out on technical manuals and so it is not clear that the corpora are comparable in difficulty. It is also possible that the original salience weights are more suited for the domain of manuals and the optimal setting of weights for our BNC segment would be worth investigating.

The Lappin and Leass algorithm did not perform significantly differently with the two sets of GRs, although Buchholz' GR finder outperformed Briscoe and Carroll's parser in the GR evaluation. However, as neither of the parsers was trained on the BNC, it is not clear whether a comparison of the anaphora resolution algorithm using the two parsers with the parsers' grammatical relation performance (evaluated on Susanne) is meaningful.

5 Conclusion

Primarily, we have implemented a version of the Lappin and Leass anaphora resolution algorithm which only relies on grammatical relations, without losing information encoded in a full parse. The extraction of GRs is more robust, thus our implementation makes the Lappin and Leass algorithm more robust without a decrease in performance.

The performance of the anaphora resolution algorithm did not appear to decrease when different intermediate parsers were employed. The lack of difference in performance supports the results of our earlier work, where we employed various full parsers in the anaphora resolution algorithm (relying on the full parser structure), with little effect on accuracy.

Having shown that using different state-of-the-art parsers has little effect on performance, we can thus conclude that future effort in anaphora resolution should be directed towards improving the basic resolution algorithm.

Acknowledgements

We would like to thank Sabine Buchholz for the output of her GR finder. Our thanks also go to Joe Hurd for reading previous drafts of this paper.

References

C. Barbu and R. Mitkov. 2001. Evaluation tool for rule-based anaphora resolution methods. In *Proceedings of the 39th ACL/10th EACL*, pages 34–41.

E. J. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1499–1504.

S. Buchholz, J. Veenstra, and W. Daelemans. 1999. Cascaded grammatical relation assignment. In P. Fung and J. Zhou, editors, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, pages 239–246.

S. Buchholz. 2002. *Memory-Based Grammatical Relation Finding.* Ph.D. thesis, University of Tilburg.

J. Carroll, E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 447–454.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proceedings of the 4th ACL/SIGDAT Workshop on Very Large Corpora*, pages 14–27.

W. Daelemans. 1996. Abstraction considered harmful: Lazy learning of language processing. In H. J. van den Herik and A. Weijiters, editors, *Proceedings of the Sixth Beligian-Dutch Conference on Machine Learning*, pages 3–12.

D. Elworthy. 1994. Does Baum-Welch reestimation help taggers? In *Proceedings of the 4th Conference on Applied NLP*, pages 53–58.

H. Gaifman. 1965. Dependency systems and phrase–structure systems. *Information and Control*, 8(3):304–337.

D. G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

C. Kennedy and B. Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 113–118.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

S. Lappin and M. McCord. 1990. A syntactic filter on pronominal anaphora for slot grammar. In *Proceedings of ACL*, pages 135–142.

G. Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

M. Marcus, R. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguisitcs*, 19(2):313–330.

J. Preiss. 2002. Choosing a parser for anaphora resolution. In *Proceedings of DAARC*, pages 175–180.

G. Sampson. 1995. *English for the computer.* Oxford University Press.

J. Veenstra and A. van den Bosch. 2000. Single-classifier memory-based phrase chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning (CoNLL) and the Second Learning Language in Logic Workshop (LLL)*, pages 157–159.