

# A General-Purpose, off-the-shelf Anaphora Resolution Module: Implementation and Preliminary Evaluation

Massimo Poesio      Mijail A. Kabadjov

University of Essex, Department of Computer Science  
Wivenhoe Park, Colchester, CO4 3SQ, UK  
poesio@essex.ac.uk, malexa@essex.ac.uk

## Abstract

GuiTAR is an anaphora resolution system designed to be modular and usable as an off-the-shelf component of a NL processing pipeline. We discuss the system's design and a preliminary evaluation of the two algorithms implemented in the current version of the system – for definite descriptions and for pronoun resolution.

## 1. Introduction

Although there is a growing interest in using anaphora resolution (AR) modules as components of the processing pipeline for applications such as information extraction (Gaizauskas and Humphreys, 2000), question answering (Morton, 2000; Watson et al., 2003), and summarization (Boguraev and Kennedy, 1997; Alonso and Fuentes, 2003), and although a number of robust, domain-independent algorithms for resolving pronouns, definite descriptions, and demonstratives have been proposed (Hobbs, 1978; Kennedy and Boguraev, 1996; Mitkov, 1998; Vieira and Poesio, 2000; Ng and Cardie, 2002; Byron, 2002), no domain-independent anaphora resolution module is currently available that developers of NLE applications can pick off the shelf in the way of tokenizers, POS taggers, parsers, or Named Entity classifiers. Yet it is clear that engineering one's anaphora system presents many advantages, not the least of which is that such a system could be evaluated in a task-oriented fashion, by measuring its contribution to the overall performance of a particular application. We are developing such a tool, GuiTAR (General Tool for Anaphora Resolution) as part of our research on segmentation and summarization. In this paper we briefly discuss the architecture and implementation of the system, as well as some preliminary evaluation results.

## 2. Architecture and Implementation

GuiTAR has been designed to be as independent as possible from the specifics of the modules used to extract certain information from the text – e.g., POS-taggers and parsers – and to be as modular as possible, allowing for the possibility of replacing specific components (e.g., the pronoun resolution component). These two goals are achieved in part by designing clear interfaces between the modules, in part by making the code modular.

In this section we discuss the architecture and implementation of both GuiTAR proper and of additional modules, also implemented, that compose a pipeline that can be used to run the system over raw text. The pipeline, illustrated in Figure 1, takes as an input either XML or raw text and produces an XML file annotated with anaphoric links, and an evaluation of AR performance with reference to an annotated corpus. XML is also used

as the data interchange format between modules<sup>1</sup>. GuiTAR proper (the Anaphora Resolution Module) is designed to take as input syntactic information in XML format (MAS-XML). GuiTAR augments its input with anaphoric links, leaving the rest of the input untouched. The overall architecture of the pipeline is compliant with the guidelines for AR systems set out by Byron and Tetreault (1999). GuiTAR may be used either as part of the pipeline (data-level integration) or within an application via its API.

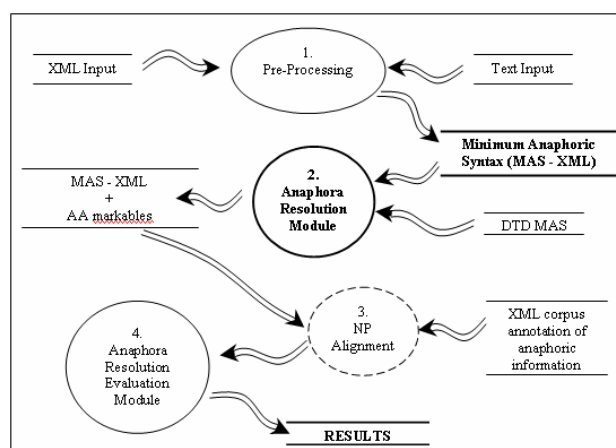


Fig. 1: Dataflow model of the processing pipeline.

### 2.1. Minimum Anaphoric Syntax (MAS-XML)

The GuiTAR XML input/output interface, MAS-XML, based on the GNOME mark-up scheme (Poesio, 2004), is meant to specify the minimal information required by anaphoric resolvers, is easy to produce from the output of a full parser, and maybe (relatively) easy to approximate starting from the output of POS taggers.

#### GuiTAR's Input

Nominal expressions are the main processing units of an anaphora resolution system. GuiTAR expects NEs to be marked with an <ne> tag and to be uniquely identifiable. GuiTAR also expects <ne> elements to have a CAT attribute (specifying the NP type: the-np, pronoun, etc)

<sup>1</sup> For the advantages of working within the SGML paradigm for NLP, see (McKelvie, Brew and Thompson 1997).

and PER, NUM and GEN attributes specifying agreement features. (Systems that cannot extract such features could either include ‘guessers’ in their preprocessors or should supply underspecified values.) The internal structure of NEs – in particular, modifiers and heads – should be marked using <mod> and <nphhead> tags. MAS-XML also requires tokens to be marked with a <w> tag, with a P attribute specifying the part-of-speech, and sentences to be marked with an <s> tag. For example, here is how the NP “the fragile eggs” would be marked-up in MAS-XML:

```
(1) <ne id="ne139" cat="the-np" per="per3" gen="neut"
    num="plur">
    <W ... P="DT">The</W>
    <mod id="m89" type="preadj">
      <W ... P="JJ">fragile</W>
    </mod>
    <nphhead>
      <W P="NNS" C="">eggs</W>
    </nphhead>
  </ne>
```

MAS-XML does not specify requirements on NE analysis. For instance, complex NEs may be either marked as single <ne> elements containing embedded <ne>s, or as flat <ne>s (the structure delivered by most partial parsers).

#### GuiTAR's Output

GuiTAR leaves its input markup intact, but adds markup information specifying that a particular anaphoric expression has been identified as belonging to the same coreference chain as another NE.<sup>2</sup> This information is expressed by separate <ante> elements, as in the MATE/GNOME markup scheme (Poesio, Bruneseaux & Romary, 1999; Poesio, 2004). An <ante> element has attributes CURRENT specifying the anaphoric expression and REL specifying a semantic relation; it contains one or more <anchor> elements specifying the antecedent.

```
(2) <ante current="ne139" rel="ident">
    <anchor antecedent="ne112" />
  </ante>
```

## 2.2 Pre-processing

This module is akin to the “Translation layer” of Byron and Tetreault (1999), and for simplicity we regard it as one module, but it actually comprises an open set of (sub)modules – one for every different input format that needs to be handled by GuiTAR. Both a text-to-MAS-XML and an example AnyXML-to-MAS-XML preprocessing modules have been implemented. The first module can be used to run GuiTAR over unrestricted text, and is based on the LT-XML suite of tools developed by the University of Edinburgh’s LTG (Brew et. al., 2000) augmented by a heuristic-based syntactic information

extractor<sup>3</sup>. The second preprocessing module is used to run the system over the GNOME corpus (see below).

## 2.3. Anaphora Resolution (GuiTAR proper)

GuiTAR is implemented in Java. Considerable effort has been made to achieve a modular design using APIs, so that different resolution algorithms for different types of anaphoric expressions can be tested, as well as algorithms that deal with all types of anaphoric expressions (e.g., (Ng and Cardie, 2002)) can be incorporated. The main classes of the implementation are illustrated in Fig. 2.

GuiTAR processes its MAS-XML compliant input incrementally left-to-right, simultaneously updating its discourse model when new <s> and <ne> elements are encountered, and using the discourse model to interpret the new NEs. The construction and update of the discourse model takes place via methods of the *DiscourseModel* API, thus isolating the system from the details of the implementation. The API makes a discourse model an instance of the class *DiscourseModelImpl*, containing one or more discourse segments, instances of the abstract class *DiscourseSegment*. (Depending on the implementation of this class, discourse segments may be simply linearly ordered, or also hierarchically embedded.) A discourse segment contains one or more utterances, instances of the class *Utterance*; these in turn contain a list of forward looking centers (Grosz, Joshi and Weinstein, 1995), instances of the abstract class *Cf*, which are realizations in a particular utterance of a given discourse entity, an instance of the *DE* class.

In order to make the system independent from specific anaphora resolution algorithms, the implementation details of specific AR algorithms are hidden behind the *AnaphoraResolution* API, whose main method is the abstract method *resolveAnaphor()* in class *Cf*. Depending on the implementation of the *Cf* class, a single interpretation algorithm may be used for all NPs, or separate algorithms may be used, e.g., for pronouns, definite descriptions, and demonstratives. In the current version of the system, an implementation of the MARS pronoun resolution algorithm (Mitkov, 2002) is provided as implementation of the *resolveAnaphor()* method for pronouns, and a partial implementation of the algorithm for resolving definite descriptions proposed in (Vieira and Poesio, 2000) as an implementation of the method for definite descriptions. No methods are included for dealing with demonstratives or proper nouns. However, implementations of generic approaches resolving all types of anaphors (e.g., (Ng and Cardie, 2002)) could also be implemented and associated with the general class *NominalGroup*.

## 3. Preliminary Experiments

Syntactic information plays an important role not only in pronoun resolution, but also in the resolution of definite descriptions, the presence of modification being

<sup>2</sup> At present the system can only resolve anaphoric expressions whose antecedents are discourse entity realized by an NE. The markup method could easily be extended to encode the results of ellipsis resolution, but it would be more difficult to use to encode the antecedents of references to abstract objects (see (Byron, 2002)).

<sup>3</sup> This heuristic component extracts NP type (see section 3), agreement features, NP head and (pre)modifiers.

one of the best signals for identifying non-anaphoric DDs (Poesio and Vieira, 1998). However, as pointed e.g., by Mitkov (2002), some of the best-published results in anaphora resolution were achieved after a substantial amount of pre- and post-processing. One of the issues we have been addressing in the early development of GuiTAR is the difference in performance between running the system over texts in which sentences and NEs have been hand-identified and over unrestricted text<sup>4</sup>. We ran two experiments over the same corpus to address these questions.

### 3.1 The corpus

For the development and evaluation of GuiTAR we have been using the GNOME corpus (Poesio, 2004), a corpus featuring texts from three different domains annotated with anaphoric relations. In this work we used texts from the museum and pharmaceutical domains. These texts contain 3354 nominal expressions.

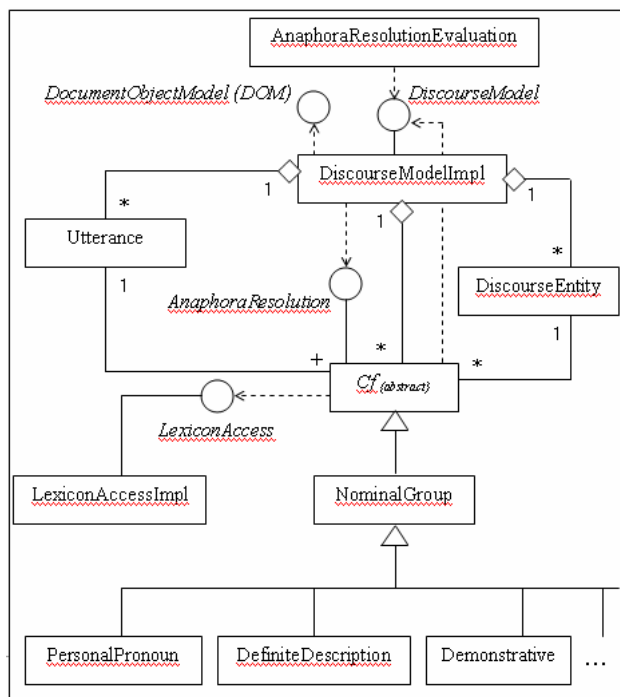


Fig. 2: GuiTAR’s Class Diagram in UML.

The nominal expressions are divided into 28 mutually exclusive categories (types); the five most frequent types are “bare-np”, “the-np”, “pers-pro”, “pn” and “a-np” representing 22.2%, 16.6%, 9.66%, 9.54%, 8.02% of the total. The focus of our work so far has been the processing of the-the-np (definite descriptions) and pers-pro (personal pronouns), which are the second and third most frequent type of NP in our corpus.

10 types of semantic relations are annotated in the corpus. At the moment we are only concerned with the identity relation, which is also the most common, representing 55.95% of all the anaphoric relations annotated (2075 relations); the rest are bridging references (Poesio et al, 2002). 21 NP types are involved in an

identity relation, the three most prominent being pers-pro, poss-pro and the-np (26.5%, 17.4%, 15.33% accordingly of a total of 1161 instances of identity relation). 95% of all the personal pronouns (pers-pro) and 32% of all the definite descriptions (the-np) are used anaphorically via an identity relation.

### 3.2 Results with hand-annotated text

In these experiments, GuiTAR was run over the annotated corpus described in section 3.1, and its performance evaluated against the annotation. The results are summarised in Table 1.

Algorithm	c	t	r	nm	wm	sm	P (%)	R (%)	F (%)
DDs (NP-type by heuristics: original POS)	115	180	175	47	18	42	65.7	63.9	64.8
DDs (NP-type by heuristics)	121	180	180	42	17	42	67.2	67.2	67.2
DDs (NP-type from corpus)	120	180	177	44	16	41	67.8	66.7	67.2
PersPro (agreement features by heuristics)	204	305	275	39	62	9	74.1	66.9	70.3
PersPro baseline (agr. features by heuristics)	159	305	275	39	107	9	57.8	52.1	54.8
PersPro (agr. features from corpus)	232	305	298	17	56	10	77.8	76.0	76.9
PersPro baseline (agr. features from corpus)	188	305	298	17	100	10	63.0	61.6	62.3

Table 1: Performance for hand-marked sentences and NPs.

Column t indicates the total number of anaphors of each type; r those identified by the system; c those correctly resolved. The errors were classified into three categories: no matches (nm), spurious matches (sm) and wrong matches (wm). (see table 1 and 2). Minimising the “no matches” improves recall; minimising the spurious matches means precision goes up; and finally, resolving wrong matches would enhance both precision and recall.

#### Definite Descriptions

Most of the “no matches” for this type of anaphora are NEs with a different head from the corresponding antecedent or containing a synonymous pre-modifier(s):

- (3) a. “the Getty Museum’s *microscope*<sup>m</sup>...the *instrument*<sup>m</sup>”
- b. “the native *British*<sup>b</sup> ... the native *Britons*<sup>b</sup>”
- c. “the *right amount*<sup>a</sup> ... the *correct amount*<sup>a</sup> of cream ...”

Endowing the system with lexical knowledge would alleviate this problem (Vieira and Poesio, 2000; Poesio et al, 2002).

The “spurious matches” are mostly cases in which a non-anaphoric definite occurs near an NE with the same head. Adding to the system discourse-new detection techniques like those in the original system (Vieira and Poesio, 2000) might help in this case. Finally, the “wrong matches” are particularly common in the pharmaceutical texts, in which many entities with the same head occur, and often the closest antecedent is not the correct one, as in the following example:

- (4) “The *patch*<sup>p</sup> ... only one Menorest *patch*<sup>p</sup>... the *patch*<sup>p</sup>”

#### Pronouns

We did not spend much time tuning the pronoun resolution algorithm; a thorough tuning of the parameters used to compute the “antecedent indicators” (Mitkov, 2002) may lead to better results. The results shown in table 2 were obtained with the following antecedent indicators turned off: indicating verbs, collocation pattern preference, immediate reference and term preference. We haven’t

<sup>4</sup> Eventually the performance of chunkers vs. full parsers is to be measured as well.

implemented the last two indicators, but by turning on “indicating verbs” and “collocation pattern preference” the number of correctly resolved pronouns slightly dropped (i.e. to 198).

### 3.3 Unrestricted Text

In these experiments the system was over the raw text of the files in the corpus, and its performance again measured against the annotation. The results are summarised in Table 2.

Algorithm	c	t	r	nm	wm	sm	P (%)	R (%)	F (%)
DDs	97	180	179	-	-	-	54.2	53.9	54.0
PersPro	192	305	293	-	-	-	65.5	63.0	64.2
PersPro Baseline	151	305	293	-	-	-	51.5	49.5	50.5

Table 2: Performance for unrestricted text

The breakdown of errors into the three classes previously identified proved to be much harder for this case, so we have not included it. In this experiment the criterion for classifying an anaphor as correctly resolved is broader than the one applied in the previous experiment. When processing the text versions of the corpus files, a process of NP alignment takes place, but there is a many-to-many relationship between the set of hand-marked NPs and the NPs identified by the partial parser. Hence, evaluating the performance of the system, the result is considered correct whenever the NE proposed as antecedent is among those included in the set of NEs matching the one hand-annotated.

We see from table 2 that for pronoun resolution the results are comparable to those obtained in the first experiment, but for DDs there is a considerable drop in performance. The overall results are broadly comparable with the results of systems participating in MUC-7. We identified three main problems, all due to problems with chunking: 1) the chunker is not able to identify DDs embedded within a possessive NE - e.g., “the king” in “the king's mistress” - so these DDs are not identified; 2) the chunker treats appositions as separate NPs<sup>5</sup>: “the French king” in “the French king Louis XIV”; 3) Incorrect chunking due to tagging errors - e.g., “The Getty Museum's microscope still\_NN works\_NNS”, where “works” is actually a verb.

## 5. Conclusion and Future work

The system has also been evaluated in terms of its contribution to segmentation; the results will appear in a future paper. Current work includes developing an improved discourse-new detector. We plan to make GuiTAR available through the OpenNLP initiative within the year.

## Acknowledgements

This research was supported in part by the Mexican Council for Science and Technology (CONACYT).

## References

- Alonso, L. and M. Fuentes, 2003. Integrating Cohesion and Coherence for Text Summarization. In *Proc. of the EACL Student Session*, pages 1–8, Budapest.
- Boguraev, B. and C. Kennedy, 1997. Saliency-based content characterisation of text documents. In *Proc. of ACL/EACL Workshop on Summarization*, pages 2–9.
- Brew, C. and D. McKelvie and R. Tobin and H. Thompson and A. Mikheev, The XML Library LT XML Version 1.2, LT-XML
- Byron, D., 2002. Resolving Pronominal Reference to Abstract Entities. In *Proc. of the 40th ACL*.
- Byron, D. and J. Tetreault, 1999. A Flexible Architecture for Reference Resolution. In *Proc. of the 9th EACL*.
- Gaizauskas, R. and K. Humphreys, 2000. Quantitative Evaluation of Coreference Algorithms in an Information Extraction System. In S. Botley and T. McEnergy (eds.), *Corpus-based and Computational Approaches to Discourse Anaphora*. Amsterdam / New York: John Benjamins, pages 145–169.
- Grosz, B., J. Aravind, and W. Scott, 1995. Centering: A Framework for Modelling the Local Coherence of Discourse. *Computational Linguistics*, 21(2).
- Hobbs, J.R., 1978. Resolving pronoun references. *Lingua*, 44, pages 311–338.
- Kennedy, C. and B. Boguraev, 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proc. of the 16th COLING*, Budapest.
- Morton, T. Coreference for NLP Applications. *Proc. 38<sup>th</sup> ACL*
- Ng, V. and C. Cardie, 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Proc of ACL*
- McKelvie, D., C. Brew, and H. Thompson, 1997. Using SGML as a Basis for Data-Intensive NLP. In *Proc. of ANLP*, Washington D.C.
- Mitkov, R., 2002. *Anaphora Resolution*. Longman
- Poesio, M. and R. Vieira, 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216. Also available as Research Paper CCS-RP-71, Centre for Cognitive Science, University of Edinburgh.
- Poesio, M., F. Bruneseaux and L. Romary, 1999. The MATE meta-scheme for coreference in dialogues in multiple languages. In *Proc. of the ACL Workshop on Standards for Discourse Tagging*. Maryland.
- Poesio, M., Ishikawa, T., Sabine Schulte im Walde, and R. Vieira. 2002. Acquiring Lexical Information for Anaphora Resolution. *Proc. of LREC*
- Poesio, M., 2004. The MATE/GNOME annotation scheme for anaphora deixis, revisited. *Proc. of SIGDIAL*.
- Vieira, R. and M. Poesio, 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4).
- Watson, R., J. Preiss, and E.J. Briscoe, 2003. The Contribution of Domain-independent Robust Pronominal Anaphora Resolution to Open-Domain Question-Answering. In *Proc of Int. Symposium on Reference Resolution*, Venezia

<sup>5</sup> Appositions are not annotated as co-referential in our corpus.