Optimising text quality in generation from relational databases

Michael O'Donnell†(micko@dai.ed.ac.uk),
Alistair Knott‡(alik@hermes.otago.ac.nz),
Jon Oberlander†(jon@cogsci.ed.ac.uk),
Chris Mellish†(chrism@dai.ed.ac.uk)

† Division of Informatics, University of Edinburgh.
‡ Department of Computer Science, Otago University.

Abstract

This paper outlines a text generation system suited to a large class of information sources, relational databases. We focus on one aspect of the problem: the additional information which needs to be specified to produce reasonable text quality when generating from relational databases. We outline how databases need to be prepared, and then describe various types of domain semantics which can be used to improve text quality.

1 Introduction

As the problems of *how* we generate text are gradually solved, a new problem is gaining prominence – *where* do we obtain the information which feeds the generation. Many domain models for existing generation systems are hand-crafted for the specific system. Other systems take advantage of existing information sources.

A good information source for text generation resides in the vast number of relational databases which are in use around the world. These resources have usually been provided for some reason other than text generation, such as inventory management, accounting, etc. However, given that the information is on hand, it can be of value to connect these databases to text generation facilities.

The benefits include natural language access to information which is usually accessed in tabular form, which can be difficult to interpret. Natural Language descriptions are easier to read, can be tailored to user types, and can be expressed in different languages if properly represented.

This paper outlines the domain specification language for the ILEX text generation system, (for *I*ntelligent *L*abelling *Ex*plorer).[1]

ILEX is a tool for *dynamic browsing* of database-defined information: it allows a user to browse through the information in a database using hypertext. ILEX generates descriptions of database objects on the fly, taking into account the user's context of browsing. Figure 1 shows the ILEX web interface, as applied to a museum domain, in this case the Twentieth Century Jewellery exhibition at the the National Museum of Scotland.[2] The links to related database objects are also automatically generated. ILEX has been applied to other domains, including personnel (Nowson, 1999), and a sales catalogue for computer systems and peripherals (Anderson and Bradshaw, 1998).

One of the advantages of using NLG for database browsing is that the system can keep track of what has already been said about objects, and not repeat that information on later pages. Appropriate referring expressions can also be selected on the basis of the discourse history. The object descriptions can be tailored to the informational interests of the user. See Knott et al. (1997) and Mellish et al. (1998) for more information on these aspects of ILEX.

In section 2, we consider some systems related to the ILEX system. Section 3 describes the form of relational database that ILEX accepts as input. Section 4 outlines what additional information – domain semantics – needs to be provided for coherent text production from the database, while section 5 describes additional information which can be provided to improve the quality of the text produced.

2 Related Work

It should be clear that the task we are discussing is very distinct from the task of response generation in a natural language interface to a database (e.g., see Androutsopoulos et al. (1995)). In such systems, the role of text planning is quite simple or absent, usually dealing with single sentences, or in the most complex systems, a single sentence answer with an additional clause or two of supporting information.

ILEX is not a query response generation system, it is an object description system. It composes a full text, at whatever size, with the goal of making that text a coherent discourse.

[1] Earlier ILEX papers have been based on Ilex 2.0, which was relatively domain-dependent. This paper is based around version 3.0 of ILEX, a re-draft to make the system domain-independent, and domain acquisition far easier. The ILEX project was supported by EPSRC grant GR/K53321.

[2] The authors thank the museum for making their database available.

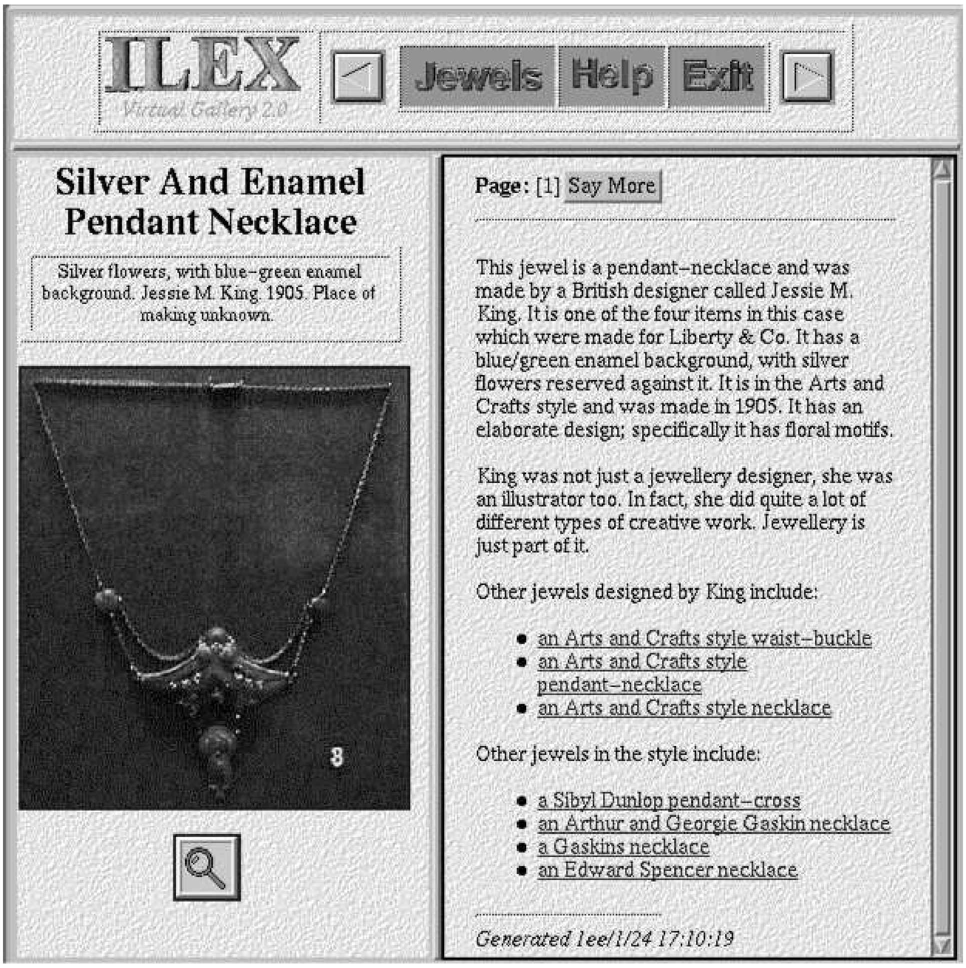

Figure 1: Browsing Object Descriptions

In this regard, ILEX should be more fruitfully compared with text generation systems such as GOSSIP (Carcagno and Iordanskaja, 1993), PEBA (Milosavljevic, 1997; Milosavljevic, 1999), or POWER (Dale et al., 1998), systems which build an extended text from an underlying database.

ILEX 3.0 has been developed to be domain independent, to handle relational databases from any domain, as long as the information is provided in the required format. The first two of the systems above are single domain systems. The third, POWER, is an extension of PEBA to handle a new domain. It is not clear however whether the resulting system is itself domain-dependent or not.

This last system is perhaps the best comparison for the ILEX system, since it also generates descriptions of museum objects from an underlying database. In that paper, the main focus is on the problem of extracting out usable information from badly structured databases (as often provided by museums), and on generating texts using only only this information (plus some linguistic knowledge). The present paper differs from this approach by assuming that information is already available in a normalised relational database. We observe, as do Dale et al. (1998), that texts generated from this information alone are quite poor in quality. We go one step further by examining what additional information can be provided to improve the quality of the text to a reasonable level.

The ILEX system has been implemented to be flexible in regards to the available domain information. With a bare minimum, the system provides poor quality texts, but as the domain developer extends the domain semantics, the quality of texts improves, up to a point where users sometimes mistake ILEX-generated texts for human-authored texts.

3 The Structure of a Relational Database

Databases vary widely in form, so we have assumed a fairly standard relational database format.

3.1 Entity Files

The database consists of a number of *entity files*, each file providing the records for a different entity type. Each record (row) in the entity file defines a unique entity. The columns define attributes of the entities. In a museum domain, we might have an entity file for museum artifacts, another for people involved with the artifacts (designers, owners, etc.), another for locations, etc. See figure 2 for a sample entity file for the Jewellery domain. Given the wide range of database formats available, ILEX assumes a tab-delimited format for database files.

ILEX imposes two requirements on the entity files it uses:

1. *Single field key*: while relational databases often use multiple attributes to form a unique key (e.g., name and birthdate), ILEX requires that each entity have a unique identifier in a single attribute. This identifier must be under a field labelled *ID*.
2. *Typing of entities*: ILEX depends strongly on a type system. We require that each entity record provides a type for the entity in a field labelled *Class*.

Some other attribute labels are reserved by the system, allowing ILEX to deal intelligently with them, including *Name*, *Short-Name* and *Gender*.

3.2 Link Files

In some cases, an entity will have multiple fillers of an attribute, for instance, a jewellery piece may be made of any number of materials. Entity files, with fixed record structure, cannot handle such cases. The standard approach in relational databases is to provide a *link file* for each case where multiple fillers are possible. A link file consists of two columns only, one identifying the entity, the other identifying the filler (the name of the attribute is provided in the first line of the file, see figure 3).

We are aware that the above specification represents an impoverished view of relational databses. Many relational databases provide far more than simple entity and link files. However, by no means all relational databases provide more than this, so we have adopted the lowest common denominator. Most relational databases can be exported in a form which meets our requirements.

3.3 Terminology

In the following discussion, we will use the following terminology:

- *Predicate*: each column of an entity file defines a *predicate*. *Class, Designer* and *Date* are thus predicates introduced in figure 2. Each link file also defines a predicate.
- *Record*: each row of an entity table provides the attributes of a single entity. The row is termed a *record* in database terminology.
- *Fact*: each entry in a record defines what we call a *fact* about that entity.[3] A fact consists of three parts: its *predicate* name, and two arguments, being the entity of the record, and the filler of the slot.
- *ARG1*: the first argument of a fact, the entity the fact is about.
- *ARG2*: the second argument of a fact, the filler of the attribute for the entity.

4 Specifying the Semantics of the Database

A database itself says nothing about the nature of the contents of each field in the database. It might be a name, a date, a price, etc. Similarly for the field label: the field label names a relation between the entity represented by the record and the entity represented by the filler. However, without further specification, we do not know what this relationship entails, apart from the label itself, e.g., 'Designer'.

Before we can begin to process a database intelligently, we need to define the 'semantics' of the database. This section will outline how this is done in the ILEX case. There has been some work on automatic acquisition of database semantics, such as in the construction of taxonomies of domain entity types (see Dale et al. (1998) for instance). However, it is difficult to perform this process reliably and in a domain-independent manner, so we have not attempted to in this case. The specification of domain semantics is still a manual process which has to be undertaken to link a database to the text generator.

To use a database for generation, additional information of several kinds needs to be provided:

1. *Taxonomic organisation*: supplying of types for each database entity, and organisation of these types into taxonomies;
2. *Taxonomic lexification*: specifying how each domain type is lexified;
3. *Data type of attribute fillers*: telling the system to expect the filler of a record slot to be an entity-id, a string, a date, etc.
4. *Domain type specification*: specifying what domain type the slot filler can be assumed to be.

Each of these aspects of domain specification will be briefly described below.

[3] Excepting the first column, which provides the entity-id for the record.

ID	Class	Designer	Date	Style	Place	Sponsor	
J-997	brooch	King01	1905	Art-Deco	London	Liberty01	
J-998	necklace	King01	1906	Art-Deco	London		
J-999	necklace	Chanel01	1910	Art-Noveux	Paris		
etc.							

Figure 2: A Sample from an Entity file

Entity	Material
J-997	silver
J-997	enamel
J-997	gold

Figure 3: A Sample from a Link file

```
(def-basic-type
  :domain jewellery-domain
  :head jewellery
  :um-link 3D-PHYS-OBJECT)

(def-taxonomy
  :type jewellery
  :subtypes (neck-jewellery wrist-jewellery
       pin-jewellery pendant buckle
       earring earring-pair finger-ring
       ringset watch button dress-clip
       hat-pin))
```

Figure 4: Defining Taxonomic Knowledge

4.1 Taxonomic Organisation

ILEX requires that the entities of the domain are organised under a domain taxonomy. The user defines a basic type (e.g., *jewellery*), and then defines the sub-types of the basic-type, and perhaps further sub-classification. Figure 4 shows the lisp forms defining a basic type in the jewellery domain, and the sub-classification of this type. The basic type is also mapped onto a type (or set of types) in the concept ontology used for sentence generation, a version of Penman's Upper Model (Bateman, 1990). This allows the sentence generator to reason about the objects it expresses.

Taxonomic organisation is important for several reasons, including among others:

1. *Expressing Entities*: each type can be related to lexical items to use to express that type (e.g., linking the type *brooch* to a the lexical item for "brooch". If no lexical item is defined for a type, a lexical item associated with some super-type can be used instead. Other aspects of the expression of entities may depend on the conceptual type, for instance pronominalisation, deixis (e.g., mass or count entities), etc.

2. *Supporting Inferences and Generalisations*: ILEX allows the user to assert generalisations about types, e.g., that Arts and Crafts jewellery tends to be made using enamel (see section 5.4). The type hierarchy is used to check whether a particular generalisation is appropriate for any given instance.

The earlier version of ILEX, Ilex2.0, allowed the full representational power of the Systemic formalism for representing domain taxonomies, including cross-classification, and multiple inheritance (both disjunctive and conjunctive). However, our experiences with non-linguists trying to define domain models showed us that the more scope for expression, the more direction was needed. We thus simplified the formalism, by requiring taxonomies to be simple, with no cross-classification or multiple inheritance. We felt that the minor loss of expressivity was well balanced by the gain in simplicity for domain developers.

4.2 Type Lexification

To express each database entity, it is essential to be able to map from its defined type, to a noun to use in a referring expression, e.g., *this brooch*.

Ilex comes with a basic lexicon already provided, covering the commonly occurring words. Each entry defines the syntactic and morphological information required for sentence generation. For these items, the domain developer needs to provide a simple mapping from domain type to lexical item, for instance, the following lisp form specifies that the domain type *location* should be lexified by the lexical item whose id is *location-noun*:

```
(lexify location location-noun)
```

For those lexical items not already defined, the domain developer needs to provide in addition lexical item definitions for the nouns expressing the types in their domain. A typical entry has the form shown in figure 5.

```
(def-lexical-item
  :name professor-noun
  :spelling "professor"
  :grammatical-features (common-noun count-noun)
)
```

Figure 5: A Sample Lexical item Specification

```
(defobject-structure jewellery
  :class :generic-type
  :subclass :generic-type
  :designer :entity-id
  :style :entity-id
  :material :generic-type
  :date :date
  :place :string
  :dimension :dimension)
```

Figure 6: Specifying Field Semantics

4.3 Data Type of Slot Fillers

Each field in a database record contains a string of characters. It is not clear whether this string is an identifier for another domain entity, a string (e.g., someone's surname), a date, a number, a type in the type hierarchy, etc.

ILEX requires, for each entity file, a statement as to how the field fillers should be interpreted. See figure 6 for an example.

Some special filler types have been provided to facilitate the import of structured data types. This includes both *:date* and *:dimension* in the current example. Special code has been written to convert the fillers of these slots into ILEX objects. Other special filler types are being added as needed.

4.4 Domain Type of Slot Fillers

The *def-predicate* form allows the domain developer to state what type the fillers of a particular field should be. This not only allows for type checking, but also allows the type of an entity to be inferred if not otherwise provided. For instance, by asserting that fillers of the *Place* field should of type *city*, the system can infer that "London" is a city even if London itself has no database record. See figure 7.

```
(def-predicate Place
  :arg1 jewellery
  :arg2 city
)
```

Figure 7: Specifying Predicate Fillers

```
(def-predicate Class
   ...
   :expression (:verb be-verb)
)
```

Figure 8: Simple Fact Expression

4.5 Summary

With just this much semantics specified, ILEX can generate very poor texts, but texts which convey the content of the database records. In the next section, we will outline the extensions to the domain semantics which are needed to improve the quality of the text produced by ILEX.

5 Extending Domain Semantics for Improved Text Quality

So far we have discussed only the simplest level of domain semantics, which allows a fairly direct expression of domain information. ILEX allows the domain developer to provide additional domain semantics to improve the quality of the text.

5.1 Expression of Facts

Unless told otherwise, ILEX will express each fact in a simple regular form, such as *The designer of this brooch is Jessie M. King*, using a template form[4]:

```
The <predicate> of <entity-expression>
   is <filler-expression>.
```

However, a text consisting solely of clauses of this form is unnatural, and depends on the predicate label being appropriate to the task (labels like *given-by* will produce nonsense sentences).

To produce better text, ILEX can be told how to express facts. The domain developer can provide an optional slot to the *def-predicate* form as shown in figure 8. The expression specification first of all defines which verb to use in the expression. By default, the ARG1 element is mapped onto the Subject, and the ARG2 onto the Object. Default values are assumed for tense, modality, polarity, voice, finiteness, quantification, etc., unless otherwise specified. So, using the above expression specification, the Class fact of a jewel would be expressed by a clause like: *This item is a brooch.*

To produce less standard expressions, we need to modify some of the defaults. A more complex expression specification is shown in figure 9, which would result in the expression such as: *For further information, see Liberty Style Guide No. 326*:

[4] ILEX3.0 borrowed this use of a default expression template from the POWER system (Dale et al., 1998). In previous versions of ILEX, all facts were expressed by full NLG as explained below.

```
(def-predicate Bib-Note
  :arg1 jewellery
  :expression (
    :adjunct1 "for further information"
    :mood imperative
    :verb see-verb
    :voice active)
)
```

Figure 9: More Complex Fact Expression

The expression form is used to construct a partial syntactic specification, which is then completed using the sentence generation module of the WAG sentence generator (O'Donnell, 1996).

With the level of domain semantics specified so far, ILEX is able to produce texts such as the two below, which provides an initial page describing database entity BUNDY01, and then a subsequent page when more information was requested (this from the Personnel domain (Nowson, 1999)):

- **Page 1**: *Alan Bundy is located in room F1, which is in South Bridge. He lectures a course called Advanced Automated Reasoning and is in the Institute for Representation and Reasoning. He is the Head of Division and is a professor.*
- **Page 2**: *As already mentioned, Alan Bundy lectures Advanced Automated Reasoning. AAR is lectured to MSc and AI4.*

This expression specification form has been designed to limit the linguistic skills needed for domain developers working with the system. Given that the domain developers may be museum staff, not computational linguists, this is necessary. The notation however allows for a wide range of linguistic expressions if the full range of parameters are used.

5.2 User Adaption

To enable the system to adapt its content to the type of user, the domain developers can associate information with each predicate indicating the system's view of the predicate's interest, importance, etc., to the user. This information is added to the *def-predicate* form, as shown in figure 10.

The user annotations allowed by ILEX include:

1. *Interest*: how interesting does the system judge the information to be to the user;
2. *Importance*: how important is it to the system that the user reads the information;
3. *Assimilation*: to what degree does the system judge the user to already know the information;

```
(def-predicate Designer
  ...
  :variation (string 1)
  :scale nominal
)
```

Figure 11: Specifying Predicate Comparability

4. *Assimilation Rate*: How quickly does the system believe the user will absorb the information when presented (is one presentation enough?).

This information influences what content will be expressed to a particular user, and in what order (more relevant on earlier pages). Information already assimilated will not be delivered, except when relevant for other purposes (e.g., when referring to the entity). If no annotations are provided, no user customisation will occur.

The values in ILEX's user models have been set intuitively by the implementers. While ideally these values would be derived through user studies, our purpose was purely to test the adaptive mechanism, and demonstrate that it works. We leave the development of real user models for later work.

ILEX has opted out of using adaptive user modelling, whereby the user model attributes are adapted as a result of observed user choices in the web interface. We leave this for future research.

5.3 Comparisons

When describing an object, it seems sometimes useful to compare it to similar articles already seen. With small addition to the domain specification, ILEX can compare items (an extension by Maria Milosavljevic), as demonstrated in the following text:

> *This item is also a brooch. Like the previous item, it was designed by King. However, it differs from the previous item in that it is made of gold and enamel, while the previous brooch was made of silver and enamel.*

For ILEX to properly compare two entities, it needs to know how the various attributes of the entity can be compared (nominal, ordinal, scalar, etc.). Again, information can be added to the *def-predicate* for each predicate to define its scale of comparability. See Milosavljevic (1997) and (1999) for more detail. Figure 11 shows the additions for the Designer predicate. Comparisons introduce several RST relations to the text structure, including *rst-contrast, rst-similarity* and *rst-whereas.*

```
(def-predicate Designer
  ...
  :importance  ((expert 10)(default 6)(child 5))
  :interest    ((expert 10)(default 6)(child 4))
  :assimilation ((expert 0)(default 0)(child 0))
  :assim-rate ((expert 1)(default 1)(child 0.5))
)
```

Figure 10: Specifying User Parameters

```
(def-defeasible-rule
 :qv ($jewel jewellery)
 :lhs (some ($X (style $jewel $X))
          (arts-and-crafts $X)))
 :rhs (some ($X (made-of $jewel $X))
          (enamel $X)))
```

Figure 12: Specifying Generalisations

- Taxonomies	OBLIGATORY
- Lexification of Types	
- Filler Domain Type Information	
- Filler Data Type Information	
- Predicate Expression	OPTIONAL
- Comparison Information	
- Generalisations	
- User Annotations	

Figure 13: Obligatory and Optional Steps in Domain Specification

5.4 Generalisations

We found it useful to allow facts about general types of entities to be asserted, for instance, that Arts and Crafts jewellery tend to be made of enamel. These generalisations can then be used to improve the quality of text, producing object descriptions as in the following:

> *This brooch is in the Arts and Crafts style. Arts and Crafts jewels tend to be made of enamel. However, this one is not.*

These generalisations are defined using *defeasible implication* – similar to the usual implication, but working in terms of *few*, *many*, or *most* rather than *all* or *none*. They are entered in a form derived from first order predicate calculus, for instance, see figure 12 which specifies that most Arts and Crafts jewellery uses enamel.

ILEX find each instance which matches the general type (in this case, instances of type *jewellery* which have *Arts and Crafts* in the *Style* role). If the fact about the generic object has a corresponding fact on the instantial object, an *exemplification* relation is asserted between the facts. Otherwise, a *concession* relation is asserted. See Knott et al. (1997) for more details on this procedure.

6 Summary

While observing people trying to convert an earlier ILEX system to a new domain, we noted the difficulty they had. To avoid these problems, we undertook to re-implement the domain specification aspects of ILEX to simplify the task.

Towards this end, we have followed a number of steps. Firstly, we reconstructed ILEX to be domain independent, with all domain information defined in declarative resource files. This means that domain developers do not have to deal with code.

Secondly, we built into ILEX the ability to import entity definitions directly from a relational database (although with some restrictions as to its form).

A database by itself does not provide enough information to produce text. Domain semantics is required. We have provided a system of incremental specification of this semantics which allows a domain developer to hook up a dynamic hypertext interface to a relational database quickly, although producing poor quality text. Minimally, the system requires a domain taxonomy, information on lexification of types, and specification of the data type of each record field.

Additional effort can then improve the quality of text up to a quite reasonable level. The additional information can include: specification of predicate expression, and specifications supporting comparisons, user adaption, and generalisations.

Figure 13 summarises the obligatory and optional steps in domain specification in ILEX.

Simplifying the domain specification task is a necessity as text generation systems move outside of research labs and into the real world, where the domain developer may not be a computational linguist, but a museum curator, personnel officer or wine salesman. We have tried to take a step towards making their task easier.

References

Gail Anderson and Tim Bradshaw. 1998. *ILEX: The intelligent labelling explorer: Experience of Building a Demonstrator for the Workstation Domain.* Internal Report, Artificial Intelligence Applications Institute, University of Edinburgh.

I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1 (1):29–81.

John Bateman. 1990. Upper modeling: organizing knowledge for natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, Pittsburgh, June.

Denis Carcagno and Lidija Iordanskaja. 1993. Content determination and text structuring: two interrelated processes. In Helmut Horocek and Michael Zock, editors, *New Concepts in Natural Language Generation*, Communication in Artificial Intelligence Series, pages 10 – 26. Pinter: London.

Robert Dale, Stephen J Green, Maria Milosavljevic, Cécile Paris, Cornelia Verspoor, and Sandra Williams. 1998. The realities of generating natural language from databases. In *Proceedings of the 11th Australian Joint Conference on Artificial Intelligence*, Brisbane, Australia, 13-17 July.

Alistair Knott, Michael O'Donnell, Jon Oberlander, and Chris Mellish. 1997. Defeasible rules in content selection and text structuring. In *Proceedings of the 6th European Workshop on Natural Language Generation*, Gerhard-Mercator University, Duisburg, Germany, March 24 - 26.

Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. 1998. An architecture for opportunistic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada.

Maria Milosavljevic. 1997. Augmenting the user's knowledge via comparison. In *Proceedings of the 6th International Conference on User Modelling*, pages 119–130, Sardinia, 2-5 June.

Maria Milosavljevic. 1999. *Maximising the Coherence of Descriptions via Comparison.* Ph.D. thesis, Macquarie University, Sydney, Australia.

Scott Nowson. 1999. *Acquiring ILEX for a Personnel Domain.* Honours Thesis, Artificial Intelligence, University of Edinburgh.

Michael O'Donnell. 1996. Input specification in the wag sentence generation system. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux Castle, UK, 13-15 June.