CUCS-418-89

# Dictionaries for Language Generation Accounting for Co-occurrence Knowledge

Frank Smadja
Computer Science Department
Columbia University
New York NY 10027
Arpanet: smadja@cs.columbia.edu

Length of paper: 4800 words.
Area of submission: C1, natural language.*

February 20, 1989

**Abstract**

Many wording choices in English sentences cannot be accounted for on semantic or syntactic grounds. They can only be expressed in terms of co-occurrence relations. Co-occurrence knowledge has been traditionally overlooked in the past, but should be included in lexicons as it is an inherent part of the language. In this paper, we demonstrate the importance of co-occurrence knowledge for language generation and we show how to include it in computational dictionaries. Using co-occurrence knowledge in the dictionary provides the generator with the information necessary for handling many lexical decisions that were previously ignored. We focus here on the process of building the dictionary, and we show how co-occurrence knowledge can be systematically entered in lexicons. Lexical relations are first identified by a cooccurrence compiler, EXTRACT. Then, domain specific semantic information is used as a criterion for classifying them. We exemplify our approach in the banking domain and we explain how it can be used by a natural language generator.

*Keywords: Co-occurrence knowledge, lexical relations, language generation, lexical knowledge

## 1 Introduction

The fact that people prefer saying *"drink a strong tea"* to *"a powerful tea"*, and prefer saying *"drive a powerful car"* to *"a strong car"* cannot be accounted for on pure syntactic or semantic grounds. There are other kinds of constraints that need to be introduced in order to filter out such oddities when producing English. Consider the following example sentences:[1]

*(1)* ⋆ *"John offered Mary a hint"*  *(2)* *"John gave Mary a hint"*
*(3)* ⋆ *"Mary perpetrated suicide"*  *(4)* *"Mary committed suicide"*

In these sentences, *"hint"* and *"suicide"* co-occur with the verbs *"give"* and *"commit"* rather than with other verbs, the sentences are collocationally restricted. This kind of behavior is unpredictable as can be seen in sentences (5) and (6). *"Murder"*, which is closely related to *"suicide"* in terms of both meaning and syntax, co-occurs with *"perpetrate"* as well as with *"commit"*, whereas *"suicide"* only co-occurs with *"commit"*.

*(5) "John perpetrated a murder"*  *(6) "John committed a murder"*

Native English speakers would not produce sentences like (1) or (3), whereas such sentences account for many language learners errors. Similarly, computer programs should definitely be provided with this information in order to avoid such mistakes.

In addition to grammatically constrained closed class words,[2] many words we produce are mainly used for structural reasons. Their presence is required by their environment, as *"commit"* is required by *"suicide"*. The basic concept is that of a *lexical relation* or *lexical affinity* due to Saussure [49]. A lexical relation between two units of language stands for a correlation of the common appearance of in the utterances of the language. For example, *"give"* and *"hint"* are bound by a lexical relation. Similarly, there is a lexical relation between *"commit"* an *"suicide"*, *"commit"* behaves as a syntactic device operating on *"suicide"*.

In this paper, our primary goal is to bring co-occurrence knowledge to bear in language generation. For this purpose we describe how to systematically account for it in computational lexicons for language generators. Language generators generally ignore such lexical constraints and are thus unable to correctly handle *collocationally restricted* sentences, that is sentences involving

[1]The lexically incorrect sentences are marked by a ⋆.

[2]Closed class words refer to small syntactic categories, such as articles, preposition etc. In contrast, open class words are nouns, adjectives and adverbs and are therefore much more numerous. Closed class words are somehow reachable by grammar rules whereas open class words are dealt with in the lexicon [Huddleston 85].

lexical relations. When not provided with such lexical information, a language generator is forced to either consider the sentence as an idiom or require that all lexical items be present in the input. We show that, in contrast, if provided with co-occurrence knowledge, language generators can accept less specified input structures and correctly handle such sentences.

To incorporate co-occurrence knowledge into the language generation process there is a need for computational dictionaries taking it into account. This is the main topic of this paper. Including lexical relations into dictionaries actually encompasses three different activities. We use example (7) in which *"check"* and *"issue* are bound by a lexical relation to illustrate these activities.

(7) *"John issued a check"*

1. Lexical relations are identified. For this we use EXTRACT a co-occurrence knowledge compiler that produces lexical relations out of large textual corpora. A relation such as *"check-issue"* is thus identified at this stage. See Section 4.1.

2. Lexical relations are semantically interpreted. Since the process of language generation consists of transforming a logical input structure into natural language, we must be able to relate lexical relations to semantic knowledge. Such a task is impracticable in general, but can performed in a specific domain. A relation such as *"check-issue"*, is interpreted by relating it to a semantic predicate available in the considered domain. See Section 4.2.

3. Lexical relations are then entered in the lexicon in a useful and coherent way for language generators. A relation such as *"check-issue"*, will be entered in the lexical entry for *"check"* and will point toward both the verb *"issue"* and the associated semantic predicate.

In this paper, we concentrate on the second activity. We have already described the first activity previously [Smadja 88] and we briefly discuss it in Section 4.1. As for the third activity, we illustrate it by an example in Section 5.3. We give a general method of semantic interpretation of lexical relations in Section 4, and we exemplify it in the domain of banking transactions in Section 5.

## 2 Existing Models

Following the early incentive of Saussure, several attempts have been made at modeling co-occurrence knowledge. Among the models, two are prominent. The first one, due to Mel'čuk [81] is integrated as part of complete linguistic model and is based on the notion of lexical function (LF). The second one, due to Benson [86] is part of a lexicographic description of English, and is based on the notion of L-type relation. Both models formalize the space of possible lexical relations. Each LF or L-type relation stands for an abstracted lexical relations into which words can enter.

LFs and L-type relations capture a very important aspect of lexical knowledge. However, directly using them is not desirable for our purpose. Although Mel'čuk's and Benson's models are based upon semantic criteria, in both cases, several types of definitions are missing. As they stand, LFs or the L-type relations are hard to interpret formally. Moreover, both models lack necessary semantic definitions. A much finer granularity is needed, but it would yield an unwieldy complexity when confronted with unlimited domains. Section 4 explains how this can be done in restricted domains, and section 5 gives an example application in a simplified banking domain.

## 3 Dictionaries and Natural Language Generation: Previous Work

Traditionally, dictionary entries are limited to semantic and syntactic information; syntagmatic lexical relations are totally ignored. When faced with collocationally restricted sentences such as Sentence (7), surface generators usually adopt either one of the following two approaches:

1. The surface generator only accepts fully lexicalized input structures. All open class lexical items must take part in the input structure (*e.g.* [Mckeown 85]). For example, in order to generate sentence (7), *"John"*, *"check"* as well as *"issue"* are required in the input structure. If the verb is not specified but simply deduced by the generator, awkward or even wrong sentences could be produced. Examples of such sentences are, *"John charged a check"*, *"John gave a check"*, or *"John paid a check"*.

2. The generator works on a lexicon containing the whole phrase in a canned structure, as one would do for idioms. The input structure generating it is thus the logical form representing it as a whole (*e.g* [Jacobs 85], [Zernik 87]). In the case of Sentence (7) this structure would represent the act of `issuing-a-check`.

Both cases are not desirable in that none takes advantage of the special relation linking *"issue"* and *"check"*. The first solution, which is the more common, should be improved because it leads to overspecified inputs. Specifying *"issue"* as well as *"check"* introduces redundancy in the input, since the verb is syntactically required by its environment.

The second solution has several flaws. First, although such cooccurrences are not fully semantically transparent,[3] they cannot be treated as semantically atomic units and should thus allow decomposition. Encoding the whole phrase as a single unit actually hides the lexical relation between the noun and the verb. Moreover, without criteria that would decide on the opacity of such sentences, this approach would lead to lexicons swamped with canned phrases.

In contrast, we argue for the use of co-occurrence knowledge in the process of generating sentences. We claim that collocationally restricted sentences can be correctly handled, and that less specified input structures are needed.

Some attempts are currently being made at producing sentences using dictionaries with co-occurrence knowledge. Both are based on Mel'čuk's model. Iordanskaya [88] is implementing a computational model of Mel'čuk complete theory. The implementation makes extensive use of LFs and is intended as a validation of Mel'čuk's linguistic model. Nirenburg *et al* [88], also make use of LFs in the framework of their work on machine translation. In their work, LFs are used in the dictionary in order to help in the process of lexical selection. Cooccurrence constraints are not included into the grammar but are rather used by a separate lexical-selection module mixing several types of constraints (mostly encyclopedic) and weighing them in order to select the appropriate lexical item. Mel'čuk based approaches present two major drawbacks. First, LFs are very hard to use in isolation from the context into which they have been introduced. Second, dictionary incorporating LFs are not yet available in English. The work that we present in this paper attempts at solving both problems. Our model is based on the simple definition of lexical relations and we describe how to incorporate co-occurrence knowledge into dictionaries.

## 4 Compiling Lexical Relations for Dictionaries

Compiling lexical relations into dictionaries is actually divided into two stages. First, the relevant lexical relations are identified. Then, they are entered in the lexicon in a useful way for language generators. Let us describe the two stages in turn.

### 4.1 Extracting Co-occurrence Knowledge

In a previous paper [Smadja 88], we described a system EXTRACT that is a co-occurrence compiler. EXTRACT derives automatically lexical relations by performing a statistical analysis of large textual corpora. EXTRACT applies the basic definition of lexical affinities and retrieves all pairs of words whose com-

[3]An expressions is semantically transparent if it can be divided into semantic constituents [Cruse 86].

mon appearance is correlated. The retrieval process consists of the following three steps for each lexical entry, $w$:

**Scan:** Scan the whole text for each appearance of $w$.

**Extract:** For each sentence containing $w$, make a note of its collocates[4]. It has been shown that 98% of lexical relations are located within a span of five [Martin 83]. Therefore, most of $w$ collocates can be retrieved by examining its environment five words before and five words after. All collocates are stored along with their syntactic category and their frequency of appearance.

**Correlate:** The statistical distribution of the collocates of $w$ is analyzed and peaks are automatically selected. A peak is defined as a collocate of $w$ whose frequency of appearance is above $\bar{f} + k\sigma$, where $\bar{f}$ is the average frequency of appearance, $\sigma$ the standard deviation, and $k$ a factor that has to be empirically determined according to the size and nature of the corpus. Let us call $k$ the co-occurrence factor. At this point, insignificant information, *i.e.* atypical collocations have been filtered out, and what remains is an ordered set of words each sharing a lexical relation with $w$. Each word appearing on the collocate list of $w$ is bound to it by a lexical relation.

The output is a set of pairs of words sharing a lexical relation, for example the relation between *"check"* and *"issue"*. EXTRACT is fully implemented and has been tested on a 300 000 word corpus taken from the UNIX$^{tm}$[5] news net. In spite of the small size of the corpus, we have been able to make useful observations. The program has been tested on 50 nouns that appear more than two hundred times in the corpus and more than 100 lexical relations have been identified. Among them were: *"to make"* with *"decision"*, *"to send"* with *"mail"*, *"high"* with *"order"*, *"take"* with *"note"*, *"to answer"* with *"question"*, *"send"* with *"request"*, and *"take"* with *"approach"*. All these pairs of words have a co-occurrence factor above 2.5, *i.e.*, the observed frequency of common appearance of the two words is 2.5 $\sigma$ above the expected one. EXTRACT is currently being tested on a more than 2, 500, 000 words corpus taken from the archives of `The Jerusalem Post`. The corpus consists of several thousand articles that have been published recently in the newspaper.

In Table 1, the first two columns represent potential outputs of EXTRACT applied to large samples of texts related to the banking domain. The next section explains how to usefully interpret lexical relations.

## 4.2 Interpreting Lexical Relations

A single word is often connected to several semantically distinct other words. For example, the noun *"check"* co-occurs with such verbs as *"issue"*, *"draw"*,

[4]By collocate, we mean the nearby open-class lexical items.

[5]UNIX is a trademark of AT&T Bell Laboratories.

*"deposit"*, *"cancel"*, and *"bounce"*. The problem with these co-occurrences is that the verbs do not have the same meaning. Natural language generators produce sentences when provided with logical representations. In order to allow full usage of co-occurrence knowledge, lexical relations should be related to semantic knowledge before being entered in the lexicon. For example, assume there is a semantic predicate DEBIT that means *"to reduce the balance of some bank account"*. In order to correctly produce sentences such as (7), the lexical relation *"check-issue"* should be related to the predicate. To perform such an analysis, a detailed semantic description is needed. This task being impracticable in unlimited domains, we must perform it in limited domains.

For descriptive purposes, let us assume that our domain is described by both a thesaurus-like knowledge base and a set of logical predicates. The predicates describe the possible actions and the thesaurus gives the semantic organization of the domain. A particular predicate *Pred* with $n$ arguments is expressed as $Pred[LR_1, LR_2, \ldots, LR_n]$, where the $LR_i$s are the logical roles of the predicate. The knowledge base contains domain specific semantic knowledge.

To interpret the lexical relations identified by EXTRACT we use domain knowledge. In this context, interpreting a lexical relation consists of associating it with a semantic predicate. For example, the lexical relation *"check-issue"* is interpreted by being related to the predicate DEBIT, where *"check"* corresponds to a given $LR_i$. The method of semantic interpretation can be described as follows:

```
For each word $w$
  For each $x$ such that $x$ and $w$ are bound by a lexical relation
    1. Find the semantically closest predicate:
       $Pred[LR_1, LR_2, \ldots, LR_n]$
       describing the semantic relation between $x$ and $w$
    2. Determine the logical role $LR_i$, played by $w$
```

Step 1 consists of interpreting the relation $x$-$w$. It works directly on the knowledge base and looks in it for the predicate closest to $x$, that is the most specific one. For example, for the word *"issue"*, the predicate DEBIT is selected. Special care is needed for general usage verbs such as *"take"* or *"make"*. These verbs are not semantically restricted and can be used in conflicting contexts. They are duplicated in the thesaurus and can thus be used to express several possible predicates. For example, the verb *"to make"* can be used to express the act of earning money as well as the act of paying money. One says, *"to make a purchase"* as well as, *"to make money"*.

Step 2 consists of determining the logical role played by $w$ within the predicate previously identified. This information is easily retrieved from examples sentences.

We are currently working on the implementation of the above method for *noun-verbs* and *adjective-noun* lexical relations. After having been applied, this method produces interpreted lexical relations that can be entered in the lexicon. Section 5 applies this method on a simplified banking domain and presents our lexical representation with the help of an example lexical entry.

## 5 A Simplified Example: the Banking Domain

### 5.1 Description of the Domain

Consider an example in the domain of banking transactions. To simplify here, we assume a single bank account which is represented by its balance. We limit the actions that can be performed on the account to DEBIT or CREDIT. DEBIT lowers the balance, and CREDIT raises it. A logical action is expressed as

$$Pred[Agent, Amount, Form, Exchange]$$

where *Pred* is the logical predicate, in our case, either DEBIT or CREDIT, the logical roles are the following:

- *Agent* is the agent initiating the transfer of funds
- *Amount* is the amount of money transferred
- *Form* is the actual Form of transfer, such as *"cash"* or *"check"*
- *Exchange* is what is bought or sold in exchange of money, examples are *"car"* or *"insurance"*.

The set of objects involved in the domain comprises all money-transfer related words, such as, *"cash"*, *"check"*, *"credit card"*. The objects are depicted in a knowledge base describing the relationships into which they enter along with their meaning. A very common classification scheme in this domain centers around the liquidity of the assets. For example, if we classify the objects in Table 1 according to this criterion, *"cash"* and *"checks"* are quickly cashed and thus considered liquid as opposed to *"interests"* and *"loans"*. Several other properties can be considered and their combination results in a thesaurus-like knowledge base describing the considered domain. In this paper, for clarity reasons, we do not describe our model to that extent but assume that such a description is available.

### 5.2 Linking LRs with Semantic Primitives

Table 1 represents some of the words involved together with their collocates and the semantic interpretation of the relation. The first column contains the considered words, the second column contains their collocates. In the third column, for each collocate, the semantic predicate associated to the lexical

| WORD | COLLOCATES | SEMANTIC PREDICATE |
|---|---|---|
| cash | take | debit |
| | withdraw | debit |
| | deposit | credit |
| check | draw | debit |
| | issue | debit |
| | sign | debit |
| | deposit | credit |
| | cash | credit |
| credit card | charge | debit |
| loan, advance | take | debit |
| | reimburse | credit |
| payment | ⋆ make | credit, debit |
| insurance | take out | debit |
| interests | pay | debit |
| | earn | credit |
| prize | win | credit |
| tax | pay | debit |
| fees | charge | debit |
| | pay | credit |
| purchase | make | debit |
| goods | buy | debit |
| | return | credit |

Table 1: Some words with their collocates semantically tagged.

relation is given. In all these examples, the logical role played by the words is *Form*. To simplify the table, the logical roles have not been represented in it.

Table 1 has been obtained by the hand application of the method described in the previous section. As an example, consider the lexical relation *"issue-check"*. *"Issuing a check"* actually means *"to pay something by signing a check"*. Interpreting the lexical relation consists in saying that it is associated with the predicate DEBIT and that the logical role played by *"check"* is that of *Form*. Some verbs such as *"make"*, marked by a $\star$ in Table 1 can be used in several situations referring to different predicates.

Such a table gives precise information for selecting lexical items in the process of generating. The act of debiting your account can be expressed in different forms depending on how the logical roles (the $LR_i$s) are lexicalized. The choice of the verb is determined by the $LR_i$s. For instance, you *"withdraw"* cash from your account whereas you *"issue"*, *"draw"* or *"make"* a check. Similarly, you *"buy"* a car whereas you *"take out"* insurance[6] policies, or *"make"* a purchase. The semantic description of the domain, as given here doesn't distinguish among all those possibilities. The distinction is introduced at the lexical level. Let us note that, these lexical decisions are not intertwined with conceptual decisions as suggested by Danlos [87]. In contrary, both types of decisions can be successfully pipelined.

## 5.3 An Example Lexical Entry

Once this analysis has been performed, the next stage is to compile the information in the dictionary. To each lexical entry is associated a set of collocational restrictions which are entered along with their semantic interpretation. See previous section. Figure 1 gives a simplified example of a lexical entry for the word *"check"*. The formalism chosen for expressing collocational constraints is that of Functional Unification Grammars [Kay 79], because of its power of expression and its uniformity. In this formalism, lexical entries are considered as functional descriptions and can thus be used by the grammar. This allows a uniform treatment of lexis as part of grammar as indicated by Halliday [66] and later by Matthiessen [88]. The lexical entry contains usual information concerning the syntax or semantics, and it is represented by attributes CAT, LEX and SEM. In addition, the lexical entry contains co-occurrence knowledge, represented by the attribute COOK. The collocational attribute contains information on both the nature of the collocates and of their logical relation. COOK contains all verbal collocates of *"check"* and gives information on their argument structure. The attribute L-ROLE is the logical

[6]In this example an insurance policy is simply considered as something that you can buy, and has the consequence debiting your account.

role played by *"check"* in the produced sentences, and the attribute PRED is the associated predicate. For example, the verb *"issue"* is used when the input logical predicate *Pred* is DEBIT and when the logical role, L-ROLE, played by the concept associated to *"check"*, is *Form.*

$$\begin{bmatrix} \text{CAT} & = & N \\ \text{LEX} & = & \textit{"check"} \\ \text{SEM} & = & \textit{pointer to the knowledge base} \\ \text{COOK} & = & \begin{bmatrix} \text{L-ROLE} = \textit{Form} \\ \left\{ \begin{matrix} \begin{bmatrix} \text{PRED} & = & \text{DEBIT} \\ \textit{verb} & = & \textit{"issue", "draw", "make", "sign"} \end{bmatrix} \\ \\ \begin{bmatrix} \text{PRED} & = & \text{CREDIT} \\ \textit{verb} & = & \textit{"deposit"} \end{bmatrix} \end{matrix} \right\} \end{bmatrix} \end{bmatrix}$$

Figure 1: An example lexical entry for the word *"check"*.

## 5.4 Generating Collocationally Restricted Sentences

In order to exemplify our approach, an example is briefly described in this section. We restrict the actual generation to the lexicalization task and we assume a logical input of the form,

$$Pred[Agent, Amount, Form, Exchange]$$

where *Pred* $\in$ { DEBIT, CREDIT }. In a full language generation environment, constraints such as stylistic or discursive, would have to be considered. Here, since we only describe the lexicalization pattern for open class words, our presentation is restricted to logical aspects, describing the mapping between logical forms and fully lexicalized functional descriptions. This lexicalization heavily centers around the lexicon. Suppose that the generator is called with the following input structure:

$$\text{DEBIT}[John, \$100, check, insurance]$$

This structure describes the fact that John took out insurance and that for the insurance he paid $ 100 in check. There are many ways to express this in English. We suppose that, responding to other constraints, the deep generator decides to produce two sentences, the first one describing the fact that John got insurance and the second one saying that he paid with a check. For

simplicity, let us consider the lexicalization of the two simple logical forms:
(1) DEBIT[$John, \$100, check, \emptyset, \emptyset$] and
(2) DEBIT[$John, \emptyset, \emptyset, \emptyset, insurance$]
Looking at the lexical entries for *"check"* and *"insurance"* respectively, the generator would find that they have collocational restrictions corresponding to the semantic predicate DEBIT. *"Check"* triggers one of *"issue"*, *"draw"*, *"make"* and *"insurance"* triggers *"take out"*. See Table 1. The verbs selected for the logical forms (1) and (2) will thus be respectively, *"issue"* and *"take out"*.[7] At this point the generator is dealing with a fully lexicalized functional description, and would generate the following two sentences, *"John issues a check of $ 100"* and *"John takes out an insurance"*.

For choosing among several collocates the generator needs other kinds of constraints than semantics or collocational. Such constraints as politeness or stylistic constraints will be necessary. As far as semantics is concerned, all collocates associated with a semantic predicate are synonymous. More generally, incorporating co-occurrence knowledge in the process of generating adds some complexity to the generation process. The main problem to be solved is the problem of interaction between co-occurrence knowledge and other kinds of knowledge in the process of generating. For example, there is a possibility of conflicting lexical constraints. This could happen if different words associated to different $LR_i$s select non intersecting collocates. We are in the process of developing a generating scheme that will handle these interactions.

## 7 Future Work and Conclusion

We are currently compiling co-occurrence knowledge on a bigger banking domain. The extraction part is already fully implemented and uses EXTRACT as a lexical relation compiler. The semantic interpretation raises some problems with the general usage verbs such as *"take"* or *"make"*. We believe that the success of the method is in the successful use of the dictionaries it helps compile.

The main thrust of our work is to demonstrate the importance of co-occurrence knowledge in current computational works, and show that its systematic use can bring much to natural language generation. We describe in this paper a compiling method based on domain specific information. The method works in two passes, first lexical relations are identified and then they are analyzed and entered in the lexicon. The first step is done with EXTRACT our co-occurrence compiler, and the second step is done by using domain specific knowledge. In parallel, we are developing a model for language generation

[7] In this example, the choice is random. The semantic description of the considered domain is not delicate enough to distinguish among them.

that will make use of this information. Such a generator would correctly handle collocationally restricted sentences and accept less specified input structures.

## 0.1 Acknowledgments

I would like to thank Kathy McKeown for her constant help during the preparation of this paper. I also want to thank Joel Walters and the English Department at Bar Ilan University, Ramat Gan, Israel, where part of this research has been performed. Many thanks to The Jerusalem Post, for graciously giving me access to its archives.

## References

**[Benson 86]** M. Benson, E. Benson and R. Ilson, *Lexicographic Description of English.* John Benjamins Publishing Company, Philadelphia, 1986.

**[Cruse 86]** D.A. Cruse, *Lexical Semantics.* Cambridge University Press, 1986.

**[Danlos 87]** L. Danlos, *The Linguistic Basis of Text Generation.* Cambridge University Press, 1987.

**[Halliday 66]** M.A.K. Halliday, *Lexis as a Linguistic Level.* In C.E. Bazell, J.C. Catford, M.A.K Halliday and R.H. Robins (eds.), *In memory of J.R. Firth* London, Longmans Linguistics Library, 1966. pp 148-162.

**[Huddleston 84]** R. Huddleston, *Introduction to the Grammar of English.* Cambridge Textbooks in Linguistics, Cambridge University Press, 1984.

**[Jacobs 85]** P. S. Jacobs, *PHRED: A Generator for Natural Language Interfaces.* Computational Linguistics, 11:4, pp 219-243, 1985.

**[Iordanskaya 88]** L. Iordanskaya, R. Kittredge, A. Polguere, *Lexical Selection and Paraphrase in a Meaning-Text Generation Model.* Presented at the fourth International Workshop on Language Generation, Catalina Island, CA, 1988.

**[Kay 79]** M. Kay, *Functional Grammar,* in Proceedings of the 5th Meeting of the Berkeley Linguistic Society, Berkeley Linguistic Society, 1979.

**[Martin 83]** W.J.R. Martin, B.P.F. Al and P.J.G van Sterkenburg, *On the processing of a text corpus: from textual data to lexicographical information.* Lexicography: Principles and Practice, Ed. R.R.K Hartmann, Applied Language Studies Series, Academic Press, London, 1983.

**[Matthiessen 88]** C. Matthiessen, *Lexicogrammatical Choice.* Presented at the fourth International Workshop on Language Generation. Catalina Island, CA, 1988.

[**Mckeown 85**] K.R. Mckeown, *Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, England, 1985.

[**Mel'čuk 81**] I.A Mel'čuk, *Meaning-Text Models: a Recent Trend in Soviet Linguistics.* The annual review of anthropology, 1981.

[**Nirenburg 88**] S. Nirenburg *et. al.*, *Lexicon building in natural language processing.* Fifteenth International Conference of the Association for Literary and Linguistic Computing, Jerusalem, Israel, June 1988.

[**Saussure 49**] F. De Saussure, *Cours de Linguistique Générale, 4ème édition.* Librairie Payot, Paris, France, 1949.

[**Smadja 88**] F. Smadja, *Lexical Cooccurrence: The Missing link in Language Acquisition.* Fifteenth International Conference of the Association for Literary and Linguistic Computing, Jerusalem, Israel, June 1988.

[**Zernik 87**] U. Zernik, *Strategies in Language Acquisition: Learning Phrases from Examples in Context.* PhD dissertation (UCLA-AI-87-1), Computer Science Department, University of California, Los Angeles, CA, 1987.