From Corpus to Codings: Semi-Automating the Acquisition of Linguistic Features

Michael O'Donnell
GMD/Institut für Integrierte Publikations- und Informationssysteme
Dolivostraße 15
64293 Darmstadt, Germany
mick@darmstadt.gmd.de

Abstract

This paper describes a tool that facilitates the linguistic coding of corpus material, through the efficient prompting of the user for relevant categories. Linguistic features are organised in terms of an inheritance network to reduce the amount of coding effort. These codings can then be exported in a form readable by statistical packages.

1 Introduction

[1]

To perform text studies, we often need to spend significant amounts of time coding our texts – splitting them up into segments of some size, and assigning features of some kind (discourse, syntactic, etc.) to each segment. We then have the problem of re-representing the coded information in a format which can be used for statistical analysis.

Ideally, some form of automatic coding of the text will be performed, using a tagger, syntactic parser, or semantic analyser. Unfortunately, the scope of such tools is limited (both in terms of syntactic coverage and semantic depth), particularly when discoursal features are being coded.

The alternative to fully automatic coding is semi-automated coding. Over the last few years, I have been developing a software tool to semi-automating some of the processes involved in coding text. The result of this work is called the "WAG Coder", which is one module of the Workbench for Analysis and generation (WAG) system – a system for single-sentence analysis and generation (O'Donnell 1994). The program runs on Macintosh computers.

The WAG Coder uses a menu-driven, window-based interface to maximally simplify the coding task. The user is prompted with a series of linguistic alternatives (choices) from which the user chooses one. Double-clicking on one of the proffered features will record the choice. Further choices will then be presented.

The coder can be set up to code text units at any linguistic level, for instance, graphological status, discoursal features, or sociological variables. However, the user does need to provide the coding scheme, which is a statement of the features to be coded, also stating which of these features are mutually exclusive. The systemic term for a set of mutually exclusive features is a system.

[1]This work was partially supported the National Science Foundation Grant IRI-9003087.

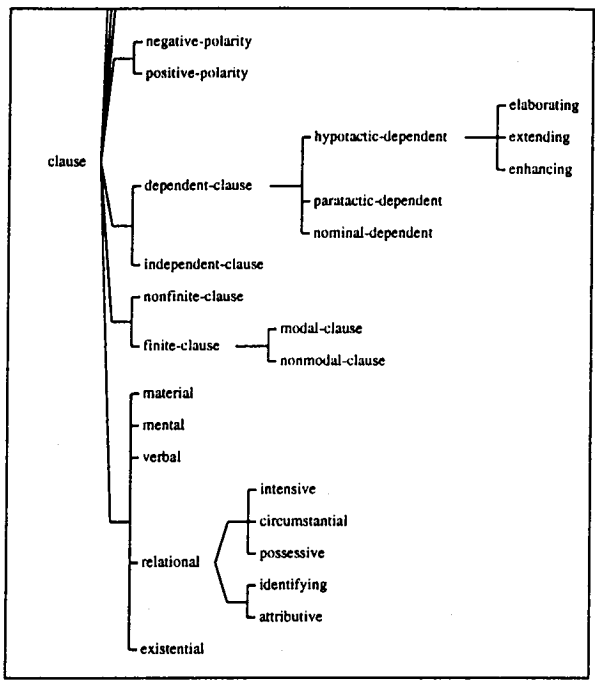

Figure 1: A Partial Graph of a Coding Scheme

It is useful to avoid coding choices which do not apply to the present unit. For instance, if we are coding an intransitive clause, it doesn't make sense to ask whether the clause is active or passive. By using a systemic network (systems organised into an inheritance network) to represent the relations between features, we avoid this problem. Some choice alternatives (systems) are made dependent on prior features being chosen. Choice sets are thus ordered in dependency.

The WAG Coder was developed under the Electronic Discourse Analyser project, funded by Fujitsu (Japan), and based in Sydney (Matthiessen *et al.* 1991). Faced with the need for grammatical profiles of our target texts, and lacking analysis tools, we developed the coder to help us build the profile. The Coder was further developed under an NSF-funded project to study the register of Newspaper articles, as part of a wider goal of making the output of a text generation system sensitive to register variation (see Bateman & Paris 1989a, 1989b; Paris & Bateman 1990).

2 Pre-Preparation

2.1 The Corpus

To prepare the corpus, the user needs to pre-segment the text, one item per line of a text file, e.g., for a study which is studying the expression of semantic events:

```
Creating a DASD dataset
This section describes the knowledge required
to create a DASD dataset.
A DASD dataset can be created
by specifying NEW in the DISP parameter
     of a DD statement.
Alternatively, the DASD dataset can be created
etc.
```

2.2 The Coding Scheme

The user must represent the coding scheme (the features in which the user is interested) in terms of a system network. This network needs to be entered into the computer in the format which is used for entering grammars in the WAG system. The input format is similar to that used in the Penman Text Generation system (WAG does in fact read Penman-format systems):

```
(defsystem
  :name congruency
  :entry-condition semantic-event
  :features (clausal-event
             nominalised-event
             adjectival-event))
```

The user provides a set of these systems, which together define a system network. These are read into the coder, which can then be used for semi-automated coding of the text corpus using this coding scheme.

The features in the coding scheme can be from any linguistic level, for instance, intonational, grammatical, semantic, speech-function, contextual (e.g., the gender of the speaker, the source of the text). These levels may be mixed freely within the coding scheme.

The user can use the Systemic Grapher, another module of the WAG system, to check that the coding scheme has been defined as intended. Figure 1 shows a part of a graph of a typical coding scheme.

3 Feature Coding

Once the text has been prepared, and the coding scheme entered, the user selects "Code Text" from a menu. A dialog window appears, with several boxes (see figure 2). The user then nominates which text file should be loaded, containing the instances to code. The interface will then present the user with each coding instance in turn (each line of text from the text file) and prompt the user to choose features for each item.

3.1 Feature selection

At the centre of the Coding window are two scrolling dialog items. One, labelled "Choice History", shows the features you have selected so far for this item (initially empty), the other showing the present choice to be made.

The second of these is labelled "Select Feature". This displays the first system in the network. If you double click on one of these choices, the feature is selected and moved to the other list the "Choice History" box. The Coder will then find the next system to the left in the system network, and present them with the choices.

In this manner, the system network is automatically traversed, the Coder prompting the user at each point. All of this proceeds in a quick and easy manner, allowing substantial amounts of instances to be coded quite quickly.

When no further choices remain, the user presses the "Store" button, which saves this coding away to a designated file. Codings can be re-accessed later for re-editing if desired.

3.2 Changing Your Mind: Deleting Features

To delete features from the "Choice History", just double-click on the relevant feature. The feature, and all the features which depend on the choice, will be removed from the Choice History.

3.3 Using Feature Defaults

Rather than stepping through each system in the coding network, the user can be presented with a dialogue window displaying all systems which are currently relevant (the condition on the system has been satisfied). See figure 3. One feature in each system is marked as the default. The user can change the default selection by clicking on one of the non-default option. When the appropriate features are selected in each system, the user presses the OK button, and the choices are recorded. This approach allows a large number of features to be coded with minimum effort, especially where most instances conform to the default coding.

4 Post-Editing of Codings

Various tools exist to view and edit codings once they have been made.

4.1 Editing Codings

The interface allows the user to call up any stored codings, and change the feature codings, comments, or text-string. From the Coder interface, you press the "View/Edit" button, and a list of all codings appears (see figure 4). Double-click on any coding, and an editor will appear. This interface also allows you to delete codings.

4.2 Filtering Codings

The "View/Edit" interface also allows you to view codings which fit a particular feature specification. Type in a feature specification (either a feature, or a logical combination of features), and only those codings which match the feature-specification will be displayed. For instance, using my coding network, I can type in any of the following feature specifications:

- **material**: Shows all material clauses in the corpus.
- **(and material abstract)**: Shows all material clauses in the Abstract stage of the text.
- **(not material)**: Shows all clauses which are not coded as material.

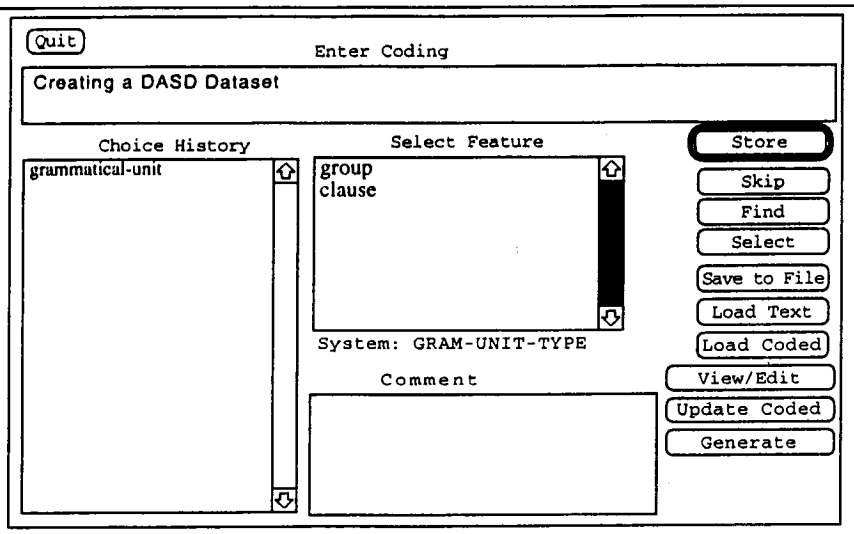

Figure 2: The Coder Window

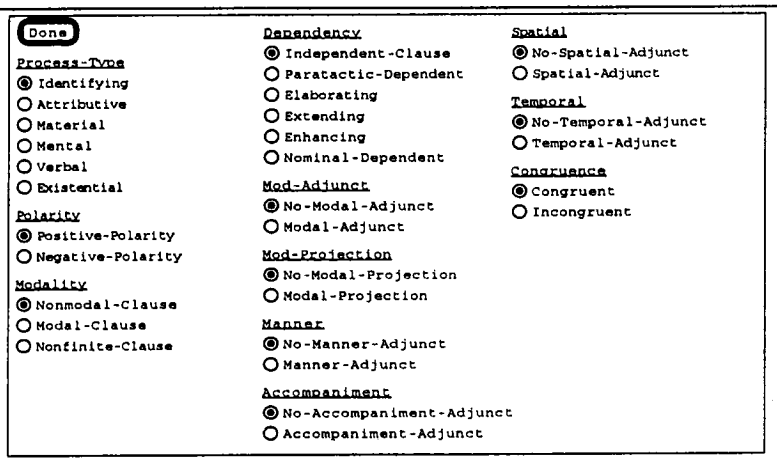

Figure 3: The Feature Defaulting Window

Feature-specifications can be arbitrarily complex, e.g. (or (not active) past). Once the feature specification is typed in, press the "Apply" button, and the restricted set of codings will be shown. If you leave the feature-specification field blank when you press the "Apply" button, then you will be presented with a list of all features. Choose one to use as the filter.

4.3 Updating Codings

If you need to change the coding scheme at any point, either changing the inheritance of categories, adding features, or adding whole systems, then the Coder allows you to update past codings without re-coding the information you already have. In the "Update Codings" mode, the coder loads up a file of saved codings, and checks the stored features against the present coding scheme. The coder will then prompt only for systems which it has no recorded feature.

5 Exporting the Data for Statistical Analysis

Coding is generally used as a first step in statistical analysis. The Coder has been designed as a module in this process. Consequently, the Coder can export the codings in a form readable by a statistical processor.

At present, tab-delimited format is supported. The user can also select which of the features are to be exported, rather than exporting all the data. In our NSF-funded register study, the exported codings are imported into the Microsoft Excel package, or into a statistical package called Statview.

Once in a statistical package, codings can be treated in two ways:

1. **unaggregated**: the codings are used as is, each coding representing one case;
2. **aggregated**: the codings are separated into text-units (e.g., by newspaper article), and feature values

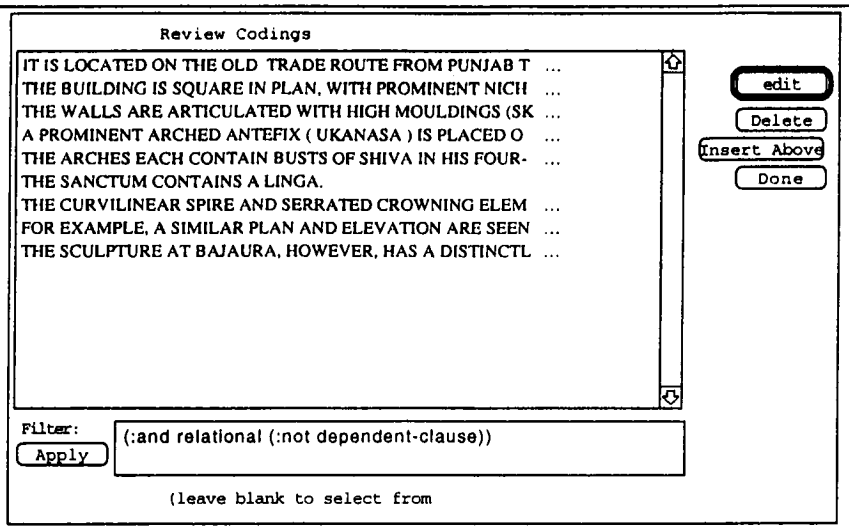

Figure 4: The Review/Edit Window

averaged out over the text-unit. The statistical data thus consists of one case per text-unit.

In the unaggregated approach, I include features for the text-type in the coding (e.g. editorial=0/1). We can then statistically analyse the relationship between these text-type features, and the other linguistic features. I typically perform the following analyses:

- Comparative Statistics: An Excel macro splits the data into subsets (those codings which include a feature *vs.* those which do not), and prints out those features whose distribution significantly differs between the subsets. For instance, for our newspaper corpus, the data was split into Editorial and non-Editorial subsets, after which the program reported the following significant differences (among others).

	Editorial	Non-Editorial
Simple-past	15%	42%
Simple-present	33%	13%
Simple-Future	8%	3%

- Correlations: Correlations between pairs of features.
- Regression Analysis: One can build a predictive model, allowing one to predict one feature based on the values of other features.

The aggregated approach offers better data for cluster analysis techniques: such techniques provide groupings of the texts, which we can then interpret as statistically-derived text-types. It remains to the analyst to label the text-types. Unfortunately, the aggregated approach requires far more coding, since we need to code a significant number of texts (rather than a significant number of, say, clauses).

6 Summary

The WAG Coder is a tool which facilitates the coding of text for use in empirical studies. It semi-automates the acquisition of features. The editing and updating of codings without total re-coding is supported. The coder allows codings to be exported in a form suitable for statistical programs.

7 Bibliography

Bateman, John & Cecile Paris 1989a "Constraining the deployment of lexico-grammatical resources during text generation: towards a computational instantiation of register theory", Technical Document, Information Sciences Institute.

Bateman, John & Cecile Paris 1989b "User Modelling and Register Analysis: A Convergence of Concerns", Technical Document, Information Sciences Institute.

Matthiessen C., O'Donnell M, and Zeng L. "Discourse Analysis and the Need for Functionally Complex Grammars in Parsing" in Proceedings of the Second Japan-Australia Joint Symposium on Natural Language Processing, October 2-5, 1991, Kyushu Institute of Technology, Iizuka City, Japan.

O'Donnell, Michael 1994 Sentence Analysis and Generation - A Systemic Perspective. Ph.D., Department of Linguistics, University of Sydney.

Paris, Cecile & John Bateman 1990 "User modeling and register theory: a congruence of concerns", Technical Report, USC/Information Sciences Institute.

Modeling Spoken Disfluencies During Human-Computer Interaction

Sharon Oviatt

Department of Computer Science and Engineering
Oregon Graduate Institute of Science & Technology
P.O. Box 91000
Portland, Oregon 97291
(oviatt@cse.ogi.edu)

This presentation summarizes several factors governing the rate of spontaneous spoken disfluencies during diverse types of human- computer interaction, and presents a predictive model accounting for their occurrence. Based on a series of empirical studies, disfluencies have been analyzed while people were engaging in unimodal and multimodal exchanges with rapidly-interactive simulated systems. In addition, disfluencies have been compared during a broad spectrum of applications entailing different communicative content, including verbal/temporal, quantitative/numeric, and cartographic/photographic descriptions.

In this line of research, spoken disfluency rates during human- computer interaction have been documented to be consistently lower than those typically observed during comparable human-human speech. Within the realm of human-computer speech, the highest disfluency rates were obtained when people spoke to map-based displays. In addition, two factors were statistically related to higher human-computer disfluency rates: (1) length of utterance, and (2) lack of structure in the presentation format. Regression techniques demonstrated that a predictive model based on utterance length alone could account for most of the variability in the rate of spoken disfluencies. Evidence is presented in support of the view that higher disfluency rates are precipitated by increased planning loads on the user.

With respect to design implications, efforts that successfully guide users' speech into briefer sentences potentially can eliminate the majority of spoken disfluencies during human-computer interaction. In this research, for example, up to 70% of all disfluent speech was eliminated entirely by using structured presentation formats. For some content, spoken input during a multimodal exchange also resulted in substantially reduced spoken disfluencies in comparison with unimodal spoken input. The long-term goal of this research is to provide a model of spoken disfluencies during human-computer interaction, as well as empirical guidance for the design of robust spoken language technology and multimodal systems.

Related References

Oviatt, S. L. Predicting spoken disfluencies during human-computer interaction, *Computer Speech and Language*, in press.

Cohen, P. R. & Oviatt, S. L. The role of voice input for human- machine communication, *Proceedings of the National Academy of Sciences*, 1995, in press.

Oviatt, S. L. (with R. A. Cole, L. Hirschman, et al.), The challenge of spoken language systems: Research directions for the nineties, *IEEE Transactions on Speech and Audio Processing*, 1995, vol. 3, no. 1, 1-21.

Oviatt, S. L., Cohen, P. R. & Wang, M. Q. Toward interface design for human language technology: Modality and structure as determinants of linguistic complexity, *Speech Communication*, European Speech Communication Association, 1994, vol. 15, nos. 3-4, 283-300.

Oviatt, S. L. & Olsen, E. Integration themes in multimodal human- computer interaction, *Proceedings of the International Conference on Spoken Language Processing*, (ed. by Shirai, Furui & Kakehi), Acoustical Society of Japan, 1994, vol. 2, 551-554.

Oviatt, S. L. Interface techniques for minimizing disfluent input to spoken language systems, *Proceedings of CHI '94*, ACM Press, Boston, Ma., 1994, 205-210.

Cohen, P. R. & Oviatt, S. L. The role of voice in human-machine communication, *Voice Communication between Humans and Machines* (ed. by D. Roe and J. Wilpon), National Academy of Sciences Press, Washington, D. C., 1994, ch. 3, 34-75.

Oviatt, S. L., Cohen, P. R. & Wang, M. Q., Reducing linguistic variability in speech and handwriting through selection of presentation format, *Proceedings of the International Symposium on Spoken Dialogue: New Directions in Human-Machine Communication*, ed. by K. Shirai, Waseda University, Tokyo, 1993, 227-230.

Oviatt, S. L., Cohen, P. R., Wang, M. & Gaston, J. A simulation-based research strategy for designing complex NL systems, *Proceedings of the ARPA Human Language Technology Workshop*, Morgan Kaufmann Publishers, Princeton, N. J., 1993, 370-375.

Oviatt, S. L., Cohen, P. R., Fong, M. W., & Frank, M. P., A rapid semi-automatic simulation technique for investigating interactive speech and handwriting, *Proceedings of the International Conference on Spoken Language Processing*, ed. by J. Ohala et al., University of Alberta, 1992, vol. 2, 1351-1354.

Oviatt, S. L. & Cohen, P. R. Discourse structure and performance efficiency in interactive and noninteractive spoken modalities, *Computer Speech and Language*, 1991, vol. 5, no. 4, 297-326.

Oviatt, S. L. & Cohen, P. R. The contributing influence of speech and interaction on human discourse patterns, in J. Sullivan and S. Tyler (eds.) *Intelligent User Interfaces*, Menlo Park, Calif.: Addison-Wesley, 1991, ch. 4, 69-83.