Processing Discourse in Dislog on the TextCoop Platform

Patrick Saint-Dizier
IRIT-CNRS, 118 route de Narbonne
31062 Toulouse Cedex France
stdizier@irit.fr

Abstract

This demo presents the TextCoop platform and the Dislog language, based on logic programming, which have primarily been designed for discourse processing. The linguistic architecture and the basics of discourse analysis in TextCoop are introduced. Application demos include: argument mining in opinon texts, dialog analysis, and procedural and requirement texts analysis. Via prototypes in the industry, this framework has now reached the TRL5 level.

1 Introduction

The TextCoop platform and the Dislog language (for Discourse in Logic) have been primarily designed for discourse processing. TextCoop was initially a research prototype which reached some maturity via its use in research, in teaching and in applications developed in cooperation with the industry. We estimate that the **TRL5 level** has now been reached. The kernel of TextCoop is now freely available under a **Creative Commons BY** licence. It is so far used by about 25 research groups or companies mainly in Europe. The foundations, the methodological elements, and the performances of TextCoop are published in (Saint-Dizier 2014). TextCoop is a platform that supports:
(1) **Dislog**, which is a language based on logic programming designed to describe in a declarative way discourse structures and the way they can be bound to form larger structures, via selective binding rules. An authoring tool has been developed to help rule specification,
(2) **an engine** associated with a set of processing strategies. This engine offers several mechanisms to deal with ambiguity and **concurrency** when different discourse structures can be recognized on a given text fragment,
(3)**a set of active constraints** that check for the well-formedness of discourse structures (e.g. precedence, dominance, co-occurence or not) which can be parameterized by the grammar writer,
(4) **input-output facilities**: the input-output streams are in XML or html formats. TextCoop can be quite directly connected to text databases or to editorial suites (e.g. Scenari), it can also, via some coding, process MS Word or Excel files,
(5) a set of **lexical resources** which are frequently used in discourse analysis (e.g. connectors),
(6) a set of about 180 **generic rules** that describe 12 frequently encountered discourse structures such as reformulation, illustration, cause, contrast, concession, etc.

2 Discourse Processing Challenges

In discourse analysis, Rhetorical Structure Theory, RST (Man et al. 1988), has been very influential in the emergence of computational models of discourse structure. RST can be used to represent the structure of e.g. explanations, reformulations, elaborations, illustrations, causes, etc. Following RST principles, almost 80 relatively general purpose relations have been introduced with various aims (http://www.sfu.ca/rst/).

In a number of 'real' texts, we observed that relations between nuclei and satellites are more complex than postulated by the RST:

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations, pages 25–28, Dublin, Ireland, August 23-29 2014.

- they may be one-to-many or even many-to-many.
- a satellite can be a nucleus for another relation,
- a nucleus and its related satellites may be non-adjacent,
- a nucleus may be linked to several satellites of different types,
- some types of satellites may be embedded into their nucleus.

This entails quite complex processing strategies and well-formedness controls that have been developed in the TextCoop platform.

Identifying discourse relations is a real challenge since linguistic cues are relatively limited. Relations are investigated together with their linguistic markers in e.g. (Delin et al. 1994), (Marcu 1997), (Miltasaki et al. 2004). They are then applied in e.g. for language generation (Rossner et al. 1992) and (Saito et al. 2006), with an extensive study on how markers can be quite systematically acquired. (Stede 2012) develops a typology of markers.

There are at the moment a few well-known and widely used language processing environments. They are essentially used for sentence processing. The reasons are essentially that the sentence level and its substructures are the crucial levels of analysis for a large number of investigations based on information extraction, opinion analysis, or machine translation. However, investigations and projects on e.g. summarization, language generation or question-answering do require an intensive discourse analysis level.

Mainly dedicated to sentence processing with some limited discourse analysis capabilities, let us note the GATE platform (http://gate.ac.uk/) which is widely used, and the Linguastream (http://www.linguastream.org) system which is based on a component architecture, making the system really flexible. RST was first used in natural language generation as a powerful means to structure arguments in a coherent way. The GETARUNS system (http://project.cgm.unive.it/getaruns.html), based on the LFG grammar approach, has some capabilities to process discourse structures and argumentation. (Marcu 1997, 2000) developed a large and robust discourse analyzer for the purpose of automatic summarization. Finally the Hilda system is a discourse parser based on a support vector machine classification (Hernault et al. 2010).

3 Main Features of Dislog and TextCoop

Dislog rule system extends the possibilities offered by regular expressions. Rules are composed of terminal, preterminal and non-terminal symbols (used to encode grammars specific to a phenomenon: e.g. temporal expressions). Symbols are associated with feature structures. The language allows optionality and iterativity markers over non-terminal and preterminal symbols. Dislog also allows 'gap' symbols, which are symbols that stand for finite sequences of words of no interest for the rule which must be skipped. Dislog offers the possibility to specify in a gap a list of elements which must not be skipped: when such an element is found before the termination of the gap, then the gap fails. Finally, rules are associated with a pattern that allows the construction of a representation based on XML tags or on dependencies.

As an illustration consider the simple rules for the 'advice' structure as expressed in technical texts:

Advice → verb(V,pref,infinitive), gap(G), punctuation(P)./

[it,is], adv(prob), gap(G1), exp(advice1), gap(G2), eos./ exp(advice2), gap(G), eos.

Resources: verb(pref): choose, prefer, ...

exp(advice1): a good idea, better, recommended, preferable

exp(advice2): a tip, an advice, best option, alternative, etc. ... adv(prob): probably, possibly, ...

For the first element of the rule (verb + gap + punctuation) the pattern that elaborates the resulting XML structure could be written as follows: <advice> V, G, P </advice>

where V, G and P are respectively the strings of words corresponding to the verb, the gap and the punctuation, as specified in the rule. In that case, the result is:

<advice> It is better to mention the capacity of high bandwidth probes </advice>, because these can be used for advanced tests.

Besides these relatively standard features, Dislog and TextCoop have the following original features

which are of much interest for discourse processing and application development:

- Dislog allows the use of **knowledge (e.g. ontologies) and reasoning procedures** via a specific field in the rules. Reasoning can be used e.g. to resolve analysis ambiguities or to elaborate a semantic representation. Predefined reasoning predicates are included, additional ones can be encoded by the grammar writer.

- Dislog has two types of rules, which basically have the same format: rules to recognize discourse structures as illustrated above and **selective binding rules** whose goal is to bind discourse structures, adjacent or not, to form larger units (e.g. an illustration with what is illustrated, an argument conclusion with its support(s), a goal-title and its instructions and warnings in a procedure, or an evaluative expression and its related arguments in opinion analysis). Bounding nodes are defined that limit the distance between discourse units which can be bound.

- TextCoop offers **concurrency and synchronization mechanisms**, which can be parameterized, in order to manage conflicts or priorities among discourse relations,

- TextCoop offers a **variety of processing strategies** which may be selected by the rule author. For example, right-to-left processing is recommended when the main linguistic cues are to the right in the rule. TextCoop has a by-default left-to-right strategy.

- TextCoop also offers **active constraints that express discourse structure well-formedness constraints** (dominance, non-dominance, precedence, strict adjacence, etc.) which are checked at each step of the parsing process. These can be parameterized depending on the discourse structures.

4 The Demos

4.1 The Basics of TextCoop

We first propose to present the overall linguistic architecture of the system and the structure of discourse analysis rules. Typical rule examples will be considered so that the audience gets a clear idea of their form and how to create or update rules and corresponding lexical data. Elements of our rule authoring system will be outlined. To illustrate rules, text samples will be processed and the results explained.

The audience can also propose text fragments that the system will process according to a predefined set of discourse analysis rules, or a given set of new or updated rules produced during the demo (this can be done quite fast). Finally relevant cases where reasoning is useful to resolve ambiguities will be presented. We will focus in particular on cases where ontologies help to disambiguate the assignment of discourse structures, e.g. between elaboration and illustration.

4.2 Argument mining in opinion texts

In (Garcia et al. 2012), we show how arguments can be extracted from opinion texts, in conjunction with evaluative expressions, so that it is possible to know **why** consumers are happy or unhappy with a certain product. In this work we shown that a number of arguments are composed of a more or less explicit evaluative expression (the conclusion in argumentation theory) followed by a support, expressed by means of discourse structures. For example, in *well located hotel, close to the museums and restaurants* the positive evaluation related to the location is further supported by an illustration (or an elaboration), which indicates why it is well located. Similarly, restrictions (attacks) can be expressed by contrast or concessive relations.

We show (1) how discourse structures relevant to the expression of argument supports or attacks are expressed in Dislog and how they are recognized in texts, (2) how they are bound to their related evaluative expressions, which may not be adjacent, by means of selective binding rules.

The result is (1) opinion texts with discourse structure annotations in XML, (2) a set of marks assigned to each evaluated attribute indicating the consumer satisfaction level, and (3) a list of supports or attacks for each of these attributes. This project is now an industrial prototype. A synthesis of these supports and attacks remains an open problem.

4.3 Procedural and requirement text analysis and improvement

This is a large project (Lelie) whose aim is to help technical writers to improve the way they produce procedural texts, specifications or requirements (Barcellini et al. 2012), (Kang et al. 2013). We address here the detection of inappropriate discourse structures. The discourse structure of technical documents is analyzed (including titles and instructions). Then error diagnosis are produced (1) according to the fact that some text fragments do not follow authoring guidelines as specified by the company or (2) that sentences have very complex discourse structures which may entail ambiguities or intensive understanding efforts from operators.

In this demo, we outline (1) the recognition of discourse structures typical of technical texts (advice, warnings, instructions, titles, prerequisites, evaluations, etc. mainly explanation structures) and how they are temporally or causally connected. Then, we show (2) how error diagnosis rules can be written in Dislog and (3) how technical texts are then tagged with appropriate error diagnosis and correction recommendations. Parts of this project are an industrial prototype 'Lelie for requirements'.

4.4 Analysis of dialogue structures

The project (Farmer) on which this demo is based involves a cooperation between Dundee, Potsdam, Toulouse and Warsaw universities. Its goal is to extract arguments for or against a certain controversial issue in a dialogue. RST as well as IAT (Budzynska et al 2013) are considered as a theoretical framework.

In this demo, we show how Dislog can be used to identify (1) elementary dialogue units, (2) the illocutionary acts and forces associated with each unit and (3) the nature of the transitions between units, based on discourse and dialogue patterns. These three points are realized using Dislog rules in different manners. They will be shown and demonstrated on the BBC Moral Maze corpus.

References

Barcellini, F., Grosse, C., Albert, C., Saint-Dizier, P., Risk Analysis and Prevention: LELIE, a Tool dedicated to Procedure and Requirement Authoring, LREC, Istanbul, May 2012.

Budzynska, K., Janier, M., Reed, C., Saint-Dizier, P., Theoretical Foundations for Illocutionary Structure Parsing, CMNA, Springer, 2013.

Delin, J., Hartley, A., Paris, C., Scott, D., Vander Linden, K., 1994. Expressing Procedural Relationships in Multilingual Instructions, Proceedings of the Seventh International Workshop on Natural Language Generation, pp. 61-70,Maine, USA.

Garcia Villalba, M., Saint-Dizier, P. 2012. *A framework for extracting arguments in opinion texts*, international journal of cognitive intelligence and natural intelligence (IJCINI), IGI Global, 6(4).

Hernault, H., Prendinger, H., du Verle, D., Ishizuka, M. 2010. *HILDA: A Discourse Parser Using Support Vector Machine Classification*, Discourse and Dialogue Vol 1(3).

Kang, J., Saint-Dizier, P. 2013. Requirement Mining in Safety Documents, CMNA, Roma.

Mann, W., Thompson, S. 1988. *Rhetorical Structure Theory: Towards a Functional Theory of Text Organisation*, TEXT 8(3) pages 243-281.

Marcu, D. 1997. The Rhetorical Parsing of Natural Language Texts, ACL97.

Marcu, D, 2000. *The Theory and Practice of Discourse Parsing and Summarization*, MIT Press.

Miltasaki, E., Prasad, R., Joshi, A., Webber, B. 2004. Annotating Discourse Connectives and Their Arguments, proceedings of new frontiers in NLP.

Rosner, D., Stede, M. 1992. Customizing RST for the Automatic Production of Technical Manuals, in R. Dale, E. Hovy, D. Rosner and O. Stock eds., *Aspects of Automated Natural Language Generation*, LNAI, Springler-Verlag.

Saint-Dizier, P. 2014. *Challenges of discourse processing: the case of technical documents*, Cambridge Scholars.

Saito, M., Yamamoto, K., Sekine, S. 2006. Using Phrasal Patterns to Identify Discourse Relations, ACL06.

Stede, M. 2012. *Discourse Processing*, Morgan and Claypool Publishers.