LELIE: A Tool Dedicated to Procedure and Requirement Authoring

Flore Barcellini, Corinne Grosse
CNAM, 41 Rue Gay Lussac,
Paris, France,
Flore.Barcellini@cnam.fr

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

Abstract

This short paper relates the main features of LELIE, phase 1, which detects errors made by technical writers when producing procedures or requirements. This results from ergonomic observations of technical writers in various companies.

1 Objectives

The main goal of the LELIE project is to produce an analysis and a piece of software based on language processing and artificial intelligence that detects and analyses potential risks of different kinds (first health and ecological, but also social and economical) in technical documents. We concentrate on procedural documents and on requirements (Hull et al. 2011) which are, by large, the main types of technical documents used in companies.

Given a set of procedures (e.g., production launch, maintenance) over a certain domain produced by a company, and possibly given some domain knowledge (ontology, terminology, lexical), the goal is to process these procedures and to annotate them wherever potential risks are identified. Procedure authors are then invited to revise these documents. Similarly, requirements, in particular those related to safety, often exhibit complex structures (e.g., public regulations, to cite the worse case): several embedded conditions, negation, pronouns, etc., which make their use difficult, especially in emergency situations. Indeed, procedures as well as safety requirements are dedicated to action: little space should be left to personal interpretations.

Risk analysis and prevention in LELIE is based on three levels of analysis, each of them potentially leading to errors made by operators in action:

1. Detection of inappropriate ways of writing: complex expressions, implicit elements, complex references, scoping difficulties (connectors, conditionals), inappropriate granularity level, involving lexical, semantic and pragmatic levels, inappropriate domain style,
2. Detection of domain incoherencies in procedures: detection of unusual ways of realizing an action (e.g., unusual instrument, equipment, product, unusual value such as temperature, length of treatment, etc.) with respect to similar actions in other procedures or to data extracted from technical documents,
3. Confrontation of domain safety requirements with procedures to check if the required safety constraints are met.

Most industrial areas have now defined authoring recommendations on the way to elaborate, structure and write procedures of various kinds. However, our experience with technical writers shows that those recommendations are not very strictly followed in most situations. Our objective is to develop a tool that checks ill-formed structures with respect to these recommendations and general style considerations in procedures and requirements when they are written.

In addition, authoring guidelines do not specify all the aspects of document authoring: our investigations on author practices have indeed identified a number of recurrent errors which are linguistic or conceptual which are usually not specified in authoring guidelines. These errors are basically identified from the comprehension difficulties encountered by technicians in operation using these documents to realize a task or from technical writers themselves which are aware of the errors they should avoid.

Proceedings of the EACL 2012 Workshop on Computational Linguistics and Writing, pages 35–38,
Avignon, France, April 23, 2012.

2 The Situation and our contribution

Risk management and prevention is now a major issue. It is developed at several levels, in particular via probabilistic analysis of risks in complex situations (e.g., oil storage in natural caves). Detecting potential risks by analyzing business errors on written documents is a relatively new approach. It requires the taking into account of most of the levels of language: lexical, grammatical and style and discourse.

Authoring tools for simplified language are not a new concept; one of the first checkers was developed at Boeing[1], initially for their own simplifyed English and later adapted for the ASD Simplified Technical English Specification[2]. A more recent language checking system is Acrolinx IQ by Acrolinx[3]. Some technical writing environments also include language checking functionality, e.g., MadPak[4]. Ament (2002) and Weiss (2000) developed a number of useful methodological elements for authoring technical documents and error identification and correction.

The originality of our approach is as follows. Authoring recommendations are made flexible and context-dependent, for example if negation is not allowed in instructions in general, there are, however, cases where it cannot be avoided because the positive counterpart cannot so easily be formulated, e.g., *do not dispose of the acid in the sewer*. Similarly, references may be allowed if the referent is close and non-ambiguous. However, this requires some knowledge.

Following observations in cognitive ergonomics in the project, a specific effort is realized concerning the well-formedness (following grammatical and cognitive standards) of discourse structures and their regularity over entire documents (e.g., instruction or enumerations all written in the same way).

The production of procedures includes some controls on contents, in particular action verb arguments, as indicated in the second objective above, via the Arias domain knowledge base, e.g., avoiding typos or confusions among syntactically and semantically well-identified entities such as instruments, products, equipments, values, etc.

There exists no real requirement analysis system based on language that can check the quality and the consistency of large sets of authoring recommendations. The main products are IBM Doors and Doors Trek[5], Objecteering[6], and Reqtify[7], which are essentially textual databases with advanced visual and design interfaces, query facilities for retrieving specific requirements, and some traceability functions carried out via predefined attributes. These three products also include a formal language (essentially based on attribute-value pairs) that is used to check some simple forms of coherence among large sets of requirements.

The authoring tool includes facilities for French-speaking authors who need to write in English, supporting typical errors they make via 'language transfer' (Garnier, 2011). We will not address this point here.

This project, LELIE, is based on the TextCoop system (Saint-Dizier, 2012), a system dedicated to language analysis, in particular discourse (including the taking into account of long-distance dependencies). This project also includes the Arias action knowledge base that stores prototypical actions in context, and can update them. It also includes an ASP (Answer Set Programming) solver [8] to check for various forms of incoherence and incompleteness. The kernel of the system is written in SWI Prolog, with interfaces in Java. The project is currently realized for French, an English version is under development.

The system is based on the following principles. First, the system is parameterized: the technical writer may choose the error types he wants to be checked, and the severity level for each error type when there are several such levels (e.g., there are several levels of severity associated with fuzzy terms which indeed show several levels of fuzziness). Second, the system simply tags elements identified as errors, the correction is left to the author. However, some help or guidelines are offered. For example, guidelines for reformulating a negative sentence into a positive one are proposed. Third, the way errors are displayed can be customized to the writer's habits.

We present below a kernel system that deals

[1] `http://www.boeing.com/phantom/sechecker/`

[2] ASD-STE100, `http://www.asd-ste100.org/`

[3] `http://www.acrolinx.com/`

[4] `http://www.madcapsoftware.com/products/madpak/`

[5] `http://www.ibm.com/software/awdtools/doors/`

[6] `http://www.objecteering.com/`

[7] `http://www.geensoft.com/`

[8] For an overview of ASP see Brewka et al. (2011).

with the most frequent and common errors made by technical writers independently of the technical domain. This kernel needs an in-depth customization to the domain at stake. For example, the verbs used or the terminological preferences must be implemented for each industrial context. Our system offers the control operations, but these need to be associated with domain data.

Finally, to avoid the variability of document formats, the system input is an abstract document with a minimal number of XML tags as required by the error detection rules. Managing and transforming the original text formats into this abstract format is not dealt with here.

3 Categorizing language and conceptual errors found in technical documents

In spite of several levels of human proofreading and validation, it turns out that texts still contain a large number of situations where recommendations are not followed. Reasons are analyzed in e.g. e.g., (Béguin, 2003), (Mollo et al., 2004, 2008).

Via ergonomics analysis of the activity of technical writers, we have identified several layers of recurrent error types, which are not in general treated by standard text editors such as Word or Visio, the favorite editors for procedures.

Here is a list of categories of errors we have identified. Some errors are relevant for a whole document, whereas others must only be detected in precise constructions (e.g., in instructions, which are the most constrained constructions):

- General layout of the document: size of sentences, paragraphs, and of the various forms of enumerations, homogeneity of typography, structure of titles, presence of expected structures such as summary, but also text global organization following style recommendations (expressed in TextCoop via a grammar), etc.
- Morphology: in general passive constructions and future tenses must be avoided in instructions.
- Lexical aspects: fuzzy terms, inappropriate terms such as deverbals, light verb constructions or modals in instructions, detection of terms which cannot be associated, in particular via conjunctions. This requires typing lexical data.
- Grammatical complexity: the system checks for various forms of negation, referential forms, sequences of conditional expressions, long sequences of coordination, complex noun complements, and relative clause embeddings. All these constructions often make documents difficult to understand.
- Uniformity of style over a set of instructions, over titles and various lists of equipments, uniformity of expression of safety warnings and advice.
- Correct position in the document of specific fields: safety precautions, prerequisites, etc.
- Structure completeness, in particular completeness of case enumerations with respect to to known data, completeness of equipment enumerations, via the Arias action base.
- Regular form of requirements: context of application properly written (e.g., via conditions) followed by a set of instructions.
- Incorrect domain value, as detected by Arias.

When a text is analyzed, the system annotates the original document (which is in our current implementation a plain text, a Word or an XML document): revisions are only made by technical writers.

Besides tags which must be as explicit as possible, colors indicate the severity level for the error considered (the same error, e.g., use of fuzzy term, can have several severity levels). The most severe errors must be corrected first. At the moment, we propose four levels of severity:

ERROR Must be corrected.

AVOID Preferably avoid this usage, think about an alternative,

CHECK this is not really bad, but it is recommended to make sure this is clear; this is also used to make sure that argument values are correct, when a non-standard one is found.

ADVICE Possibly not the best language realization, but this is probably a minor problem. It is not clear whether there are alternatives.

The model, the implementation and the results are presented in detail in (Barcellini et al., 2012).

4 Perspectives

We have developed the first phase of the LELIE project: detecting authoring errors in technical documents that may lead to risks. We identified a number of errors: lexical, business, grammatical, and stylistic. Errors have been identified from ergonomics investigations. The system is now fully implemented on the TextCoop platform and has been evaluated on a number of documents. It is now of much interest to evaluate user's reactions.

We have implemented the system kernel. The main challenge ahead of us is the customization to a given industrial context. This includes:

- Accurately testing the system on the company's documents so as to filter out a few remaining odd error detections,
- Introducing the domain knowledge via the domain ontology and terminology, and enhancing the rules we have developed to take every aspect into account,
- Analyzing and incorporating into the system the authoring guidelines proper to the company that may have an impact on understanding and therefore on the emergence of risks,
- Implementing the interfaces between the original user documents and our system, with the abstract intermediate representation we have defined,
- Customizing the tags expressing errors to the users profiles and expectations, and enhancing correction schemas.

When sufficiently operational, the kernel of the system will be made available on line, and probably the code will be available in open-source mode or via a free or low cost license.

Acknowledgements

This project is funded by the French National Research Agency ANR. We also thanks reviewers and the companies that showed a strong interest in our project, let us access to their technical documents and allowed us to observed their technical writers.

References

Kurt Ament. 2002. *Single Sourcing. Building modular documentation*, W. Andrew Pub.

Flore Barcellini, Camille Albert, Corinne Grosse, Patrick Saint-Dizier. 2012. *Risk Analysis and Prevention: LELIE, a Tool dedicated to Procedure and Requirement Authoring,* LREC 2012, Istanbul.

Patrice Béguin. 2003. Design as a mutual learning process between users and designers, *Interacting with computers*, 15 (6).

Sarah Bourse, Patrick Saint-Dizier. 2012. *A Repository of Rules and Lexical Resources for Discourse Structure Analysis: the Case of Explanation Structures*, LREC 2012, Istanbul.

Gerhard Brewka, Thomas Eiter, Mirosław Truszczyński. 2011. Answer set programming at a glance. *Communications of the ACM* 54 (12), 92–103.

Marie Garnier. 2012. Automatic correction of adverb placement errors: an innovative grammar checker system for French users of English, Eurocall'10 proceedings, Elsevier.

Walther Kintsch. 1988. *The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model*, Psychological Review, vol 95-2.

Elizabeth C. Hull, Kenneth Jackson, Jeremy Dick. 2011. *Requirements Engineering*, Springer.

William C. Mann, Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organisation, *TEXT* 8 (3), 243–281. Sandra A. Thompson. (ed.), 1992. *Discourse Description: diverse linguistic analyses of a fund raising text*, John Benjamins.

Dan Marcu. 1997. *The Rhetorical Parsing of Natural Language Texts*, ACL'97.

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

Vanina Mollo, Pierre Falzon. 2004. *Auto and allo-confrontation as tools for reflective activities*. Applied Ergonomics, 35 (6), 531–540.

Vanina Mollo, Pierre Falzon. 2008. *The development of collective reliability: a study of therapeutic decision-making*, Theoretical Issues in Ergonomics Science, 9(3), 223–254.

Dietmar Rösner, Manfred Stede. 1992. *Customizing RST for the Automatic Production of Technical Manuals*, In Robert Dale et al. (eds.) Aspects of Automated Natural Language Generation. Berlin: Springer, 199–214.

Dietmar Rösner, Manfred Stede. 1994. *Generating multilingual technical documents from a knowledge base: The TECHDOC project*, In: Proc. of the International Conference on Computational Linguistics, COLING-94, Kyoto.

Patrick Saint-Dizier. 2012. Processing Natural Language Arguments with the TextCoop Platform, *Journal of Argumentation and Computation*.

Edmond H. Weiss. 2000. *Writing remedies. Practical exercises for technical writing*, Oryx Press.