Annotating Documents by their Intended Meaning to Make them Self Explaining: an Essential Progress for the Semantic Web

Hervé Blanchon and Christian Boitet

Laboratoire CLIPS
BP 53
38041 Grenoble Cedex 9, France
{herve.blanchon, Christian.boitet}@imag.fr

Abstract. A Self-Explaining Document (SED) is a document enriched with annotations keeping track of all possible interpretations with respect to a given grammar and dictionary, as well as disambiguating choices. If disambiguation is complete and has been done by the author himself, a SED conveys "the author's intention". The availability of SEDs might considerably reduce misunderstanding between authors and readers, and perhaps lead to the assignment of a "meaning certification level" to any part of a document. We present ways to integrate these annotations into an arbitrary XML document (SED-XML), and to make them visible and usable to readers for accessing the "true content" of a document. We also show that, under several constraints, a SED, once translated into a target language L, might be transformed into an SED in L with no human interaction. Hence, the SED structure might be used in multilingual as well as in monolingual contexts, without addition of human work.

1 Introduction

We first proposed the concept of Self-Explaining Document in [5] as an answer to some question raised while experimenting with a new incarnation of the interactive translation paradigm, the DBMT approach (Dialogue-Based Machine Translation) in the LIDIA project [6]. We observed (again) that translation introduces ambiguities which are not present in the source text. Traduttore, traditore… For example, the two French words "remplacer" (replace by a new thing) and "replacer" (put back into place) were both translated by "replace" in English. It also happened that all disambiguated analyses of a sentence produce the same translation, as ambiguous as the original. One example was the translation from French into Russian of the famous sentence «the man sees the girl in the park with a telescope».

This raised an objection to DBMT: what is the use of disambiguating the source text if ambiguities reappear in the translation(s), or even worse if new ones are created? Would it not be better to try and produce translations which preserve the ambiguities, and dispense with Interactive Disambiguation (ID) altogether?

Unfortunately, the experience of human translation shows that ambiguities can be exactly preserved only in some cases, and that to do it purposefully is quite difficult

and often leads to unnatural ways of expression in the translation. It is also quite clear that the "transferable" ambiguities vary with the target language. Finally, although some texts may be intentionally ambiguous, especially in poetry and politics, we take it that the vast majority of ambiguities are not intentional, but are due to the intrinsic nature of natural languages. Of course, some authors write more clearly than others, but all authors write unambiguously in any programming language, unambiguous by construction, and ambiguously in any natural language, ambiguous by nature!

This motivated the idea of Self-Explaining Documents: if the source and target documents are accompanied by their (unambiguous) linguistic structure, with the indications of potentially ambiguous parts, and if the reader in the target language may obtain a clarification of unclear parts in a user-friendly way, the objection disappears. As human users are notably not very sensitive to ambiguities, however, we should find a way to warn the reader that the target text is ambiguous.

2 DBMT: a Context for SED production

After having worked in the direction of "suboptimization" for 15 years [10], we turned to high quality Dialogue-Based Machine Translation. DBMT is a new paradigm derived from that of Interactive MT (IMT) and geared to various translation situations where other approaches, such as the Linguistic-Based (LBMT) and the Knowledge-Based (KBMT) approaches, are not adequate.

In DBMT, although the linguistic knowledge sources are still crucial, and extralinguistic knowledge might be used if available, emphasis is on indirect pre-editing through negotiation and clarification dialogues with the author in order to get high quality translations without revision.

Authors are distinguished from "spontaneous" writers or speakers by the fact that they want to produce a "clean" final message and may be willing to enter into such dialogues. The crucial difference with usual IMT is that interactive disambiguation is not performed during an analysis or transfer process, but on a "multiple" data structure factorizing all possible analysis results. Hence, the author is not "slave of the system", but decides if and when s/he wants to enter the disambiguation dialogue.

The first situation considered (in 1990) was the production of multilingual technical documentation in the form of HyperCard[1] documents. A page of such a document is a card. A card may contain different kinds of objects such as graphics, buttons and textual fields. The linguistic MT lingware was based on multilevel transfer with interlingual acceptions, properties and relations implemented in ARIANE-G5.

The first mockup, LIDIA-1 [6], demonstrates the idea on a HyperCard stack presenting short ambiguous French sentences in several disambiguating contexts. This document is translated into three documents, German, Russian and English. Although this mockup does not implement all features of the general design — a complete implementation would have called for considerably more human resources than were available — it demonstrates the potential of the approach.

[1] HyperCard is a Macintosh-based (MacOS-7 to MacOS-9) environment for the production of hypertextual documents called "stacks".

2.1 LIDIA-1 Through an Example

The author can trigger the most frequent functions through the LIDIA-1 palette. Its first line contains the LIDIA tools (process the selected object, show the treatment progress, show the annotations, and show the reverse translation). The second line contains browsing tools.

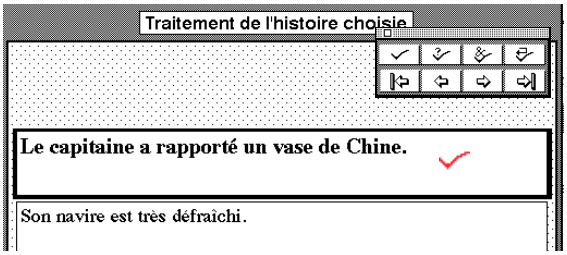

Fig. 1. Selection of a field to be translated[2]

After analysis, a button (**? !!** - **Fig. 2**) appears above the object to be translated if its content is ambiguous.

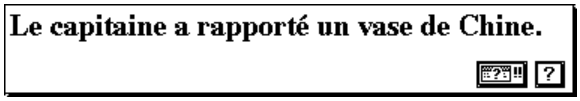

Fig. 2. Pending disambiguation questions[2]

When the author decides to disambiguate the object's content, s/he clicks on the button. Pending questions are then asked (**Fig. 3** & **Fig. 4**).

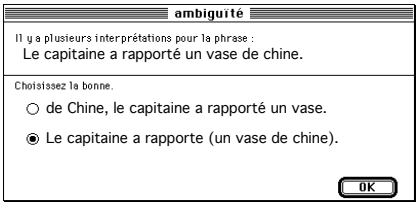

Fig. 3. Structural disambiguation question[3]

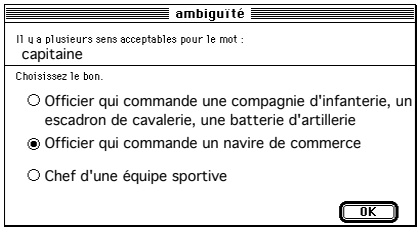

Fig. 4. Word sense question[4]

Finally, the system produces the corresponding "exact" translation. For our example sentence, we will get two different translations in German if we disambiguate

---

[2] The captain brought back a vase (form/of) China.

[3] "From China, the captain brought back a vase" *vs.* "The captain brought a Chinese vase".

[4] Sense-1: army captain; Sense-2: boat captain; Sense-3: sports team captain.

according to two different contexts: (1) “Der *Hauptmann* hat eine Vase aus China mitgebracht. Die Vase ist englisch.“, and (2) “Der *Kapitän* hat eine chinesische Vase mitgebracht. Sein Boot ist sehr verblaßt.“

2.2 SED Production in the DBMT Framework

Let us now explain how a SED can be produced. For this, some details about the LIDIA lingware architecture are needed.

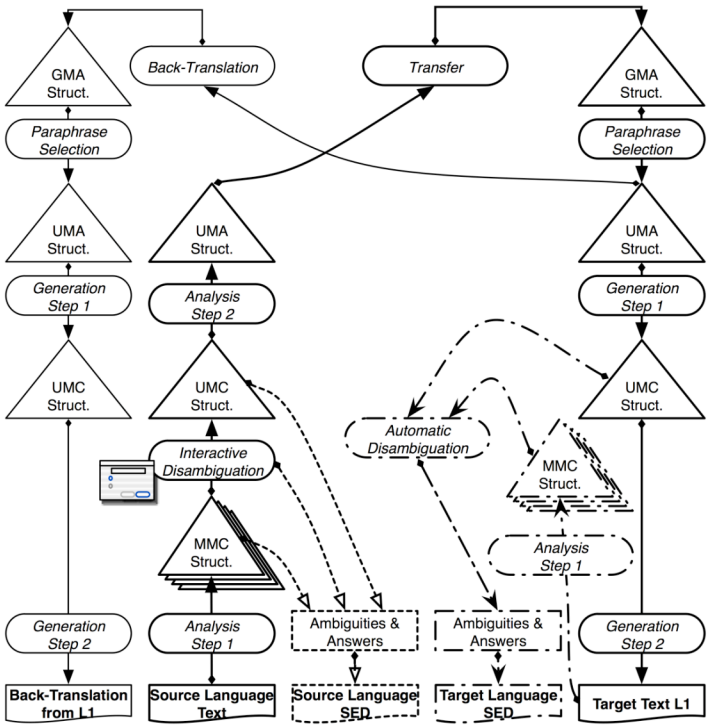

Fig. 5. DAE production workflow

Each sentence of the source text is analyzed to produce a *source-mmc* (multisolution, multilevel, concrete) structure. A representation of a text is “concrete” if the corresponding text can be recovered from it by using a standard traversal algorithm and simple morphological and graphematical generation rules. Familiar examples are textbook constituent structures and dependency structures (with left-to-right traversal of the leaves or infix traversal of all nodes). Otherwise, we say that the representation is “abstract”. Note that the information contained in both kinds of structures (on labels and other more or less complex annotations) may be of the same linguistic “depth”: there may be “deep” concrete structures and “surface” abstract structures, in this sense, although the opposite is of course more frequent.

This *source-mmc* structure is then used to produce a disambiguation question tree. Once the author has answered the questions, the system gets the non-ambiguous *source-umc* (unisolution, multilevel, concrete) structure chosen by the author.

This *source-umc* structure is abstracted into a *source-uma* (unisolution, multilevel, abstract) structure. In an abstract structure, some lexical information can be "featurized", and order would be normalized. Abstract representations of utterances are far superior to concrete representations as input and output structures of transfers in semantic transfer MT or as "lexical-conceptual structures" [7] in interlingual MT, especially between distant languages. But their relation to the corresponding utterances is not as clear, a natural consequence of abstraction.

A lexical and structural transfer component produces a *target-gma* (generating, multilevel, abstract) structure. In *gma* structures, non-interlingual linguistic levels are underspecified. If present, they are used only as reflections of corresponding surface levels in the source language, and are recomputed in the first generation phase, called "paraphrase choice".

The paraphrase selection step produces a *target-uma* structure. This structure is equivalent to the structure that would be obtained after analysis and disambiguation of the target text to be produced. The translation process ends with the syntactic and morphological generation phases.

In order to produce a SED, it suffices to keep the data structures used by the ID phase, namely: the *mmc* structure, the question tree, and the disambiguating path in it. **Fig. 5** gives a functional diagram of the workflow we just described.

2.3 LIDIA-1 Distributed Software Architecture

In the LIDIA-1 architecture, the coordination server, the redaction server, the communication server and the disambiguation server were AppleScript applets. They communicated through AppleEvent messages to coordinate their work. The coordination server scheduled the whole translation process for each translation unit. The author's workstation communicated with the translation server under the SMTP protocol.

3 SED Production within LIDIA-2

LIDIA-2 is a new Java front end for DBMT services developed since 2002. LIDIA-2 produces XML documents, from which SEDS are produced.

3.1 LIDIA-2: New Software Architecture and Environment

The LIDIA-1 implementation was tightly linked with Macintosh operating system before OS-X. The manipulated data were also very poor in terms of content (no memory of the intermediate processes is kept) and their representation (plain text files).

In LIDIA-2, all DBMT services are made available on Internet through a portable interface written in Java. We use a distributed components architecture experimented within the C-STAR [3] and NESPOLE! [8] projects of speech-to-speech translation. The basic idea is to let all the components interact through a communication server (ComSwitch), to which they are attached by a Telnet connection.

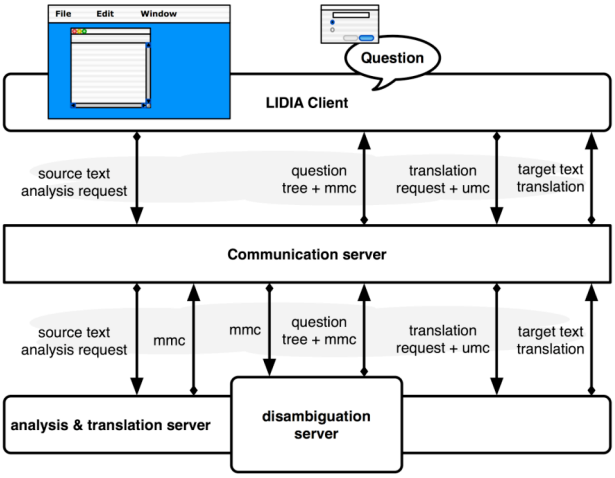

Fig. 6. LIDIA-2 architecture

LIDIA-2 appears as a text editor enhanced with DBMT functionalities. When the author has customized his environment, s/he can create a new document or open an existing one. The upper part of a document window displays its textual content, and the lower part displays information about the documents status.

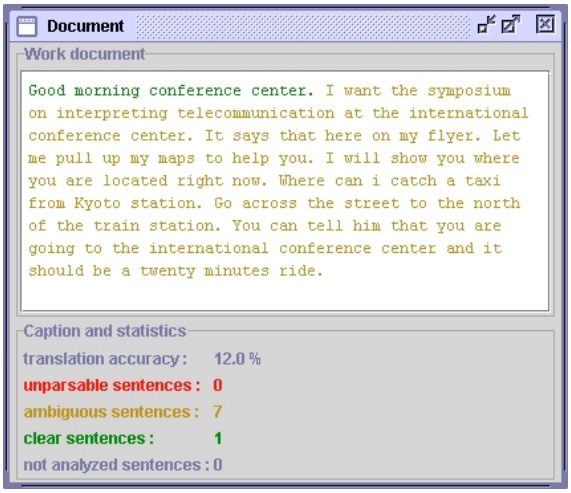

Fig. 7. Document window (after analysis step 1)

We use the ID module developed for English [2, 4]. Integration was immediate within our new architecture.

After the author has requested the analysis (**Fig. 7**), the ambiguous sentences are displayed in brown, and the unambiguous ones in green. In our example, only the first sentence is unambiguous sentence. When the author double-clicks an ambiguous sentence, the disambiguation questions tree prepared is traversed down.

For the sentence "I want the symposium on interpreting telecommunications at the international conference center", the root question (**Fig. 8**) is first asked. The Status part shows that this is the first question, and that at most, one question can follow. The author may Suspend or Reinitialize the session, or go back to the Previous question. When s/he has answered a question, s/he proceed to the Next one. When every question are answered, the author closes the session. At that point, for demonstration purposes, the sentences may be translated in French using a translation database.

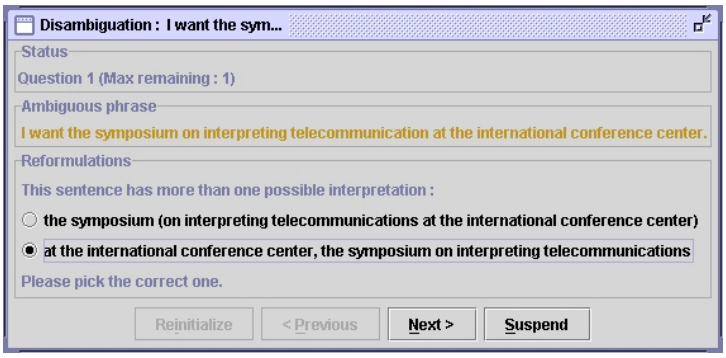

Fig. 8. Root question of the selected sentence

3.2 The LIDIA-2 Document

The XML document produced by LIDIA-2 is manipulated through the DOM API. When the author opens a document, its syntax is checked with the SAX API. The document contains a header, and its actual content. The header consists of a title and information about the author. The content is a set of paragraphs made of sentences.

```
<phrase source="ENG" stamp="51054803544695">
  <original><![CDATA[ I will show you where you are located right now.]]></original>
  <question>
    <reformulation choix="NON"><![CDATA[I will show you (where you are located right now).]]>
      <analyse><![CDATA[…]]></analyse>
    </reformulation>
    <reformulation choix="OUI"><![CDATA[right now, I will show you where you are located.]]>
      <analyse><![CDATA[…]]></analyse>
    </reformulation>
  </question>
  <traduction cible="FRA"><![CDATA[Je vais tout de suite vous montrer où vous êtes.]]></traduction>
</phrase>
```

Fig. 9. a LIDIA-2 document excerpt

Each sentence has a source language and a unique transaction identifier that allows the environment to keep track of the ongoing treatments for each sentence. The original content of the sentence, the answered question tree, and the produced translations are also represented. As far as the question tree is concerned, it stores the answer path through the different question items and the *umc* structure with its solution number associated with each terminal question.

3.3 Producing and Visualizing a DAE

After disambiguation, a source SED can be filtered out from the LIDIA-2 document. Sentences and answered disambiguation trees without the *umc* structures are kept.

As we want a SED to be a portable document, we developed a specialized viewer. Ideally, such a viewer should present the document and highlight its ambiguous segments. The reader should then be able to select any ambiguous segment and get a presentation of the author's intended meaning.

Although our first viewer, developed in Java, is fairly simple and user interaction with the document is still poor, it gives a concrete idea of what can be done.

In this first version, the ambiguous segments are not highlighted (see below): in order to gather information about the different readings, the reader must double-click on a sentence. S/he can then browse through the questions answered by the author of the document. The rephrasings chosen by the author are then highlighted.

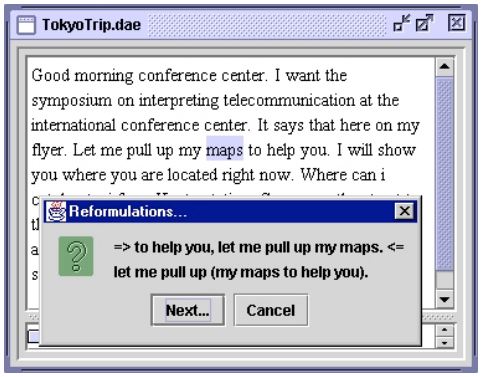

Fig. 10. Showing the author's chosen rephrasings

3.4 Outcome and Short Term Perspectives

Our first short-term goal is to make the ARIANE-G5 HTL modules for LIDIA available through the same ComSwitch as the disambiguation module. That would add new possibilities of experimentation. Going further on showing the ambiguities would require several changes in the disambiguation module itself.

The first version of the LIDIA-2 XML document structure is fairly simple, and all the information is not fully "XML-ized". For example, the *mmc* struc-

ture and the question tree use a Lisp-like representation requiring a specific module, although a DOM-based module would be more efficient and portable.

The question tree is XML-ized within the LIDIA-2 interface. It would be better to let the disambiguation module do it instead of producing a bracketed linear structure from a tree object.

Ariane-G5 does not handle XML-like data structure yet. We may implement a filter going back and forth between XML and ARIANE-G5 data structures. Such a component could be added transparently within the ComSwitch-based architecture.

4. Long Term Perspectives

The proposals we discuss now will impact on the HLT modules and/or the disambiguation module. They are thus long term goals.

4.1 Ambiguity Support and SED

4.1.1 Ambiguity as a Formal Object

In order to formalize the notion of ambiguity, let us take an example. Consider the utterance: `(1) Do you know where the international telephone services are located?`

The underlined fragment has an ambiguity of attachment, because it has two different "skeleton" [1] representations: `[international telephone] services / international [telephone services]`.

As a title, this sequence presents the same ambiguity. However, it is not enough to consider it in isolation. Take for example: `(2) The international telephone services many countries`. The ambiguity has disappeared!

It is indeed frequent that an ambiguity relative to a fragment appears, disappears and reappears as one broadens its context in an utterance. For example, in `(3) The international telephone services many countries have established are very reliable`, the ambiguity has reappeared.

From the examples above, we see that, in order to define properly what an ambiguity is, we must consider a fragment *within an utterance,* and clarify the idea that this fragment is the smallest (within the utterance) where the ambiguity can be observed.

A fragment **F** presents an ambiguity **A** of multiplicity n ($n \geq 2$) in an utterance **U** if it has n different proper representations which are all part of m ($m \geq n$) proper representations of **U**.

F is an ambiguity support if it is minimal relative to that ambiguity. This means that any strictly smaller fragment **F'** of **U** will have strictly less than n associated subrepresentations (at least two of the representations of **V** are equal with respect to **F'**).

In example `(1)` above, then, the fragment "the international telephone services", together with the two skeleton representations "the [international telephone] services / the international [telephone services]" is not minimal, because it and its two representations can be reduced to the subfragment "international telephone services" and its two representations (which are minimal).

An *ambiguity occurrence*, or simply *ambiguity*, **A** of multiplicity n (n≥2) relative to a representation system R, may be formally defined as:

A = (U, F, <S_1, S_2, …, S_m>, <f_1, f_2, … , f_n>), where m≥n and:

- **U** is a complete utterance, called the *context* of the ambiguity **A**.
- **F** is a fragment of **U**, usually, but not necessarily connex, called the **support** of the ambiguity **A**.
- **S_1, S_2, …, S_m** are proper representations of **U** in **R**, and **f_1, f_2, … , f_n** are subparts of them representing **F**.
- Minimality condition:

For any fragment **F'** of **U** strictly contained in **F**, if **f'_1**, **f'_2**, … , **f'_n** are respective parts of **f_1**, **f_2**, … , **f_n** corresponding to **F'**, then there is at least one pair **f'_i**, **f'_j** ($i \neq j$) such that **f'_i=f'_j**.

The kind of the ambiguity **A** depends on the difference between the **s_i**. It is defined in the framework of each **R**.

Fig. 11. Formal definition of ambiguity

4.1.2 Ambiguity Detection in the Current ID Process

The disambiguation methodology is based on the manipulation of tree structures. A kind of ambiguity is described with a set of patterns called an *ambiguity descriptor* (**Fig. 12**). A pattern contains variables and describes a tree structure with constraints on its geometry and labelling. Once an *ambiguity descriptor* has been recognized, a question is prepared.

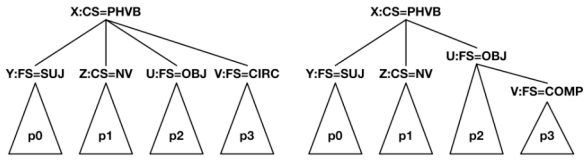

Fig. 12. An ambiguity detection beam

A question item is associated with each pattern of the beam and is produced through the manipulation, with a set of basic operators, of the values given to the variables instantiated during the recognition of the beam. The method associated with the left pattern in **Fig. 12** produces the following string: Text(**p2**), Text(**p0**) Text(**p1**) Text(**p2**). The method associated with the right pattern, produces the following string: Text(**p0**) Text(**p1**) Bracket(Text(**p2**), Text(**p3**)).

4.1.3 Towards a Better ID Preparation Process

In the current version of the French and English ID modules, within an *ambiguity descriptor*, the patterns are designed to capture the ambiguity support and some context. This context is used to produce a better labelling of the dialogue items.

For a SED, it seems to be very important to have a precise localization of the ambiguities that enable a precise highlighting of every ambiguity. It means that the ID preparation process should be able to provide the ambiguity support. They are two approaches to reach this goal.

In the first approach, the *ambiguity descriptor* patterns are not changed and the ambiguity support is associated with each *descriptor*. The support can be described in terms of the variables available in the patterns of the *descriptor*.

The second approach would change deeply the disambiguation preparation process. The new idea here is to let the patterns describe an ambiguity using only the ambiguity support. On the other hand, the rephrasing mechanism will have to pick up, in the *mmc* structure, the parts of the input text that have to be used to produce the dialogue items. It will also necessitate an embedded XML encoding of the ambiguity support allowing the GUI to clearly mark them.

4.2 Incomplete Disambiguation and Meaning Certification Levels

In the context of real applications, many ambiguities will arise, concerning word sense, attachment, argument structure, etc. Thus, there is a chance that for each sentence the question tree will be quite big, and the writer will not be willing to answer all questions, but only the most crucial ones.

Suppose that a sentence of length **N** has $\mathbf{k^N}$ interpretations and that the *ambiguity descriptors* are of average size **b**. **(k/b).N** questions in average would totally disambiguate the sentence. If, for example, **(k/b)=1/2**, there would be about 120 questions for a page of 240 words. Although answering them all may take 10 minutes or less[5] if we allow 5 seconds for each answer, the author may want to spend less time on ID.

In order to satisfy that need, answering all the questions in a question tree has not to be mandatory for the generation module to produce a translation. In other words, given an *mmc* structure, some disambiguation answers and maybe some user preferences or profile, the HLT modules have to be able to make a choice (using heuristics) and produce a unique translation, or to produce a factorized and linguistically "felicitous" representation for all remaining interpretations.

From the degree of completeness of the disambiguation of a sentence, and from the cruciality of the remaining ambiguities, it is certainly possible to compute a "meaning certification level" and associate it to the sentence. Meaning certification level can then be computed for paragraph, sections, etc., up to the whole documents, and more generally any part of it.

4.4 Target Language SED

In section 2.4, we have discussed why it would be very interesting to produce SEDs in target languages. Reaching such a goal is very demanding as far as the HLT module development is concerned because we need or each target language an "all path" analyzer which is an inverse of the generator. We intend to cooperate with other groups abroad to prototype that part.

---

[5] This has to be compared with the usual figures given by professional translators: 1 hour for each first draft translation, 20 minutes for the postedition in each target language.

5 Conclusion

The concept of SED appeared, and perhaps could only appear, in the context of our research on DBMT. That explains why, although it clearly opens many fascinating new possibilities in the use of numerical documents, no other researchers seem to work (yet!) on that concept.

However, we consider our research to be directly quite related to the larger and booming area of numerical documents and of the "semantic web" [11]. Starting from our first SED prototype, based on the new LIDIA-2 architecture, and primitive SED viewer, we now plan to contribute to the subfield of active documents [9] by building a more sophisticated GUI embedded in *a la* Thot[6]-like.environment.

References

1. Black, E., Garside, R. and Leech, G.: Statistically-Driven Grammars of English: the IBM/Lancaster Approach. Rodopi. Amsterdam (1993)
2. Blanchon, H. An Interactive Disambiguation Module for English Natural Language Utterances. NLPRS'95. Seoul, Korea, vol. **2/2**: 550-555 (1995)
3. Blanchon, H. and Boitet, C.: Speech Translation for French within the C-STAR II Consortium and Future Perspectives. ICSLP 2000. Beijing, China, vol. **4/4**: 412-417 (2000)
4. Blanchon, H. and Fais, L.: Asking Users About What They Mean: Two Experiments & Results. HCI'97. San Francisco, California, vol. **2/2**: pp. 609-912 (1997)
5. Boitet, C.: Dialogue-Based MT and self explaining documents as an alternative to MAHT and MT of controlled language. Machine Translation Ten Years On. Cranfield, England (1994)
6. Boitet, C. and Blanchon, H.: Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup. Machine Translation, vol. **9**(2): 99-132 (1995)
7. Levin, L. and Nirenburg, S. (1994) The Correct Place of Lexical Semantics in Interlingua. COLING-94. Kyoto, Japan, vol. **1/2**: pp. 349-355 (1994)
8. Metze, F., Mc Donough, J., Soltau, H., Waibel, A., Lavie, A., Burger, S., Langley, C., Levin, L., Schultz, T., Pianesi, F., Cattoni, R., Lazzari, G., Mana, N., Pianta, E., Besacier, L., Blanchon, H., Vaufreydaz, D. and Taddei, L.: The NESPOLE! Speech-to-Speech Translation System. HLT 2002. San Diego, California, USA (2002)
9. Quint, V. and Vatton, I.: Making structured documents active. Electronic Publishing Origination, Dissemination, and Design. vol. **7**(2): pp. 55-74 (1994)
10. Vauquois, B. and Boitet, C.: Automated Translation at Grenoble University. Computational Linguistics, vol. **11**(1): 28-36 (1985)
11. W3C (2001) Semantic Web. http://www.w3.org/2001/sw/.

[6] cf. http://opera.inrialpes.fr/Thot.en.html