Corpus Analysis to Extract Information

Fabrice Even
IRIN - University of Nantes
2 rue de la Houssinière
44322 Nantes, France
even@irin.univ-nantes.fr

ABSTRACT
This article presents an automatic information extraction method from poor quality specific-domain corpora. This method is based on building a semi-formal ontology in order to model information present in the corpus and its relation. This approach takes place in four steps: corpus normalization by a correcting process, ontology building from texts and external knowledge, model formalization in grammar and the information extraction itself, which is made by a tagging process using grammar rules. After a description of the different stages of our method, experimentation on a French bank corpus is presented.

Keywords
Information extraction, modeling, building ontology, poor quality corpus, corpus correction.

INTRODUCTION
This research stems from the need for information extraction from a poor quality corpus (without punctuation, with poor syntax and a lot of abbreviations). In our approach, the modeling of information needed and its identification in the corpus carry out information extraction. The modeling uses external knowledge sources to construct a semi-formal ontology, which covers a part of the corpus domain, i.e. only the knowledge actually described in corpus. This ontology is used to extract information.

After a brief presentation of different ontology-building methods, a presentation of our method is made by a description of its different stages: a corpus partial correction, ontology building from this corrected corpus and external knowledge, the ontology representation by a grammar and the information extraction engine based on this grammar. The results are evaluated and analyzed.

We work on a French corpus composed of bank texts. These texts are compilations of interviews between bank employees and clients. Our goal is to automatically extract specific information about clients (future plans, evolution of their family situation, etc.) to expand a database.

1 ONTOLOGY BUILDING METHODS
There are a lot of methods to build ontology from corpora. Most of them are based on text content. In these approaches, texts are the main source for knowledge acquisition [10]. Concepts and relations result only from a corpus analysis, without external knowledge. Aussenac-Gilles and al. [1] follow this point of view but affirm that there can be other knowledge sources than corpus. Such an approach includes two steps. The first one consists of the construction by a corpus terminological and linguistic analysis, of a first set of concepts that corresponds to terms (conceptual primitives [10]) and the extraction of lexical relations. The result of this stage is a base of primitive concepts and first concept relations (terminological knowledge base [1] & [9]). In this stage, the designer has to select terms and relations that will be modeled, those that are relevant and in the case of several meanings for one term or relation, which one must be kept. The second step is based on conceptual modeling by a study of the semantic relations between terms. This analysis gives new concept relations and new concepts, which are added to the first one. This new set of concepts and relations is also structured into a concept semantic network. An expert of the corpus domain must validate this network to express which relations are relevant (normalization [5]). The result of the entire process is a hierarchical structure of a set of terms of the domain [12]. This model is also an ontology, which can be formalized by a formal or semi-formal representation.

Using such a method is powerful for syntactically correct texts but not for poor quality corpora. The terminological step is conceivable after corpus partial correction but lexical and semantic relation extraction are impossible. Indeed linguistic tools are highly ineffective on syntactical and lexical poor corpora.

So for such poor quality corpora, our approach can be based on a terminological analysis for first concept identification but a solution other than classical methods must be found for the other modeling steps (terminological knowledge base and semantic network building).

2 CORPUS CORRECTION

The texts from our corpus are distinguished by a lot of typological errors, spelling mistakes and the use of non-standard abbreviations.

These characteristics make a correction and normalization step necessary. Indeed using the texts directly without correction gives a lot of undesirable information and causes a poor coverage of information. Therefore modeling and information extraction starts from the corrected corpus. This step concerns value formats (normalization of numbers with unit), dates (unique numeric representation and specific abbreviation treatment such days or month), abbreviation standardization (substitution of abbreviations specific to the texts or the writers by a unique one) and orthographical correction of lexical and typological errors.

This correction process is carried out with a set of contextual rules written after a lexical study of the corpus. Automata are used on the texts to applied rules.

3 DOMAIN MODEL BUILDING

According to Bachimont [3], there are no independent concepts from context or current problems, which allow building the whole knowledge of a particular domain. Ontology works like a theoretical framework of a domain and is built according to a current problem. The modeling process described here is based on this definition. Ontology is built from knowledge found in the corpus and from external knowledge (experts).

3.1 Initial ontology definition

The information searched is first informally expressed (in natural language) and next converted into predicates. These predicates are described by information patterns. This work must be done with domain experts or information extraction final users with a wide knowledge of the domain and who are able to express exactly what information must be extracted. This step gives a set of concept hierarchies. This set is the first sub-ontology (initial ontology), which is composed of predicative relations between concepts (there is a relation between two concepts when one of them is an attribute of the other). These relations are in accordance with the Attribute Consistency Postulate [8]: in each predicative relation, any value of an attribute is also an instance of the concept corresponding to that attribute.

3.2 Terminology definition

The terminology is built from the union of two sets of terms. The first is made by a terminological study of texts by linguistic tools as ANA [6]. The other is built by a set of documents about domain terminology where terms that can be used in texts are found (domain technical documents for example).

3.3 Normalization

There is not a direct correspondence between each term and a concept from initial ontology. But these concepts have to be bound to corpus terms, so a normalization process is necessary. This process takes place in three steps: initial ontology extension, terminological knowledge base (TKB) building and unification of models.

3.3.1 Initial ontology extension

Initial ontology is revised with domain experts. Some new concepts that stemmed from this first set of concepts are defined and added to hierarchy. This gives an extended initial ontology.

3.3.2 TKB building

With these same experts and domain specific documents, a new set of concepts is built from terminology: the basic concepts. From these basic concepts, others are defined recursively by inheritance. The result is a set of small hierarchies with, for each one a unique ancestor whose last heir is a basic concept. These hierarchies are normalized: each father is divided into sons by a unique criterion. They also respect the Guarino-Rigid-Property [7].

3.3.3 Models Unification

The two precedent processes give ontology linked to the current problem and a hierarchical structure linked to texts. We proceed to the unification of these two models. The extended initial ontology is unified with a hierarchy if it ancestor is a concept of the initial ontology or if a relation can be built between this ancestor and concepts from this ontology.

After these three steps, domain model is obtained, which covers all concepts relevant for information searching. An oriented graph diagram first describes this model. This model defines a semi-formal ontology because it is not dependent on a representation language [4].

4 FORMAL REPRESENTATION

To make the model usable, it is formalized into a grammar. As seen in last section, two relation types are found in this model: hierarchical relations and predicative relations. The grammar must represent these two types of relation. Also two sorts of rules are defined. On the one hand, constituent rules, which represent hierarchical relations, are defined. When we have a hierarchical relation between two concepts A and B, with B a son of A, we say that B constitutes A. On the other hand predicative rules are defined to represent predicative relations. When we have a predicative relation between two concepts C and D, D is an attribute of C (the type of attribute depends on the relation). All these rules are written with a BNF-like description.

4.1 Constituent rules

A concept C is defined by a set of rules Def(C). These rules concern terms or concepts. For each X from Def(C), X only defines C, never another concept. The notation for these rules is C ::= Def(C). There are three sorts of con-

stituent rules: select rules, conjunctive rules and disjunctives rules.

4.1.1 Select rules

The form of select rules is: C ::= B1 | B2 | ... . The concept C is defined by B1 or by B2 but not by both of them at the same time.

Examples:

<VEHICLE> ::= <CAR> | <MOTO>
| <OTHER_VEH>

<CAR> ::= ford | mercedes | …

4.1.2 Conjunctive rules

The form of conjunctive rules is: C ::= B1 + B2 + ...

The concept C is defined by a set of concepts in which all concepts are necessary to define C.

Example: <MORTGAGE> ::= <DC_LOAN> + <PROPERTY>

4.1.3 Disjunctive rules

The form of disjunctive rules is: C ::= B1 v B2 v ... . The concept C is defined by B1 or by B2 or by both of them.

Example: <PERSON> ::= <NAME> v <FIRST_NAME>

4.2 Predicative rules

These rules describe predicative concepts (also called predicate). These are concepts with attribute. Predicative relations define links between these concepts and their attributes. These rules define a predicate by one descriptor and one main attribute: the object. For a predicate, the descriptor is a unique concept, this mean that a concept cannot be the descriptor for more than one rule. The object is one of a set of possible concepts (this set is defined by the model).

These rules can have some optional attributes. These attributes give more information on the predicate but are neither necessary nor sufficient to define it.

Predicative rules are written: P ::= (descriptor = D; object = O1 | O2 | O3 | ... ; option1 = A1 | A2 | A3 | ... ; option2 = B1 | B2 | ... ; ...)

Example: PURCHASE predicate is described by figure 1.

Figure 1: PURCHASE predicate

```
<PURCHASE> ::=
(
        descriptor = <DC_PURCHASE> ;
        object = <PROPERTY>
                | <VEHICLE>
                | <BANK_PRODUCT>;
        date = <DATE>;
        amount = <SUM>
        location = <PLACE>
)
```

5 EXTRACTION ENGINE

The extraction engine is based on the grammar modeling of the domain. It proceeds in four steps: rules database creation, two tagging processes and information collection.

Rules database is composed of two sets of rules (constituent and predicative) inferred from the grammar. Constituent tagging is based on database constituent rules (each term and concept is tagged according to database rules). Second tagging is based on predicative database rules for instantiate predicates. After these step, information is collected directly. This information will expand a database.

5.1 Constituent tagging

Constituent tagging find terms, and then concepts in the text by recursive applications of constituent rules. Each time a concept is found, a tag marks it up.

Some specific concepts (with a known syntax) such as sums, rates or dates are tagged first. After that, tagging takes place in two steps: term tagging then concept propagation.

Some select rules define concepts from terms. In term tagging, these rules are applied to the corpus (for each rule C ::= t, tags of concept C mark the term t in the text). When all these rules are applied, every term in the grammar is tagged by concepts.

With concept propagation, some new concepts are found. When a rule A ::= B exists, tags of A are added to tags of B in the corpus. So concept A is marked in the texts.

Conceptual rules are applied until none of them is applicable. Then the corpus is completely tagged by constituent rules (cf. figure 2).

Figure 2: Example of predicative tagging

The text "*buy studio london in 2003*" becomes after constituent tagging:

<DC_PURCHASE>buy</DC_PURCHASE>
<PROPERTY><APARTMENT>studio
</APARTMENT><PROPERTY>
<PLACE><CITY>london</CITY></PLACE>
in <DATE>2003</DATE>

with the rules :

```
<DC_PURCHASE> ::= buy | bought
<APARTMENT> ::= studio | apartment | loft
<CITY> ::= london | paris | tokyo
<PROPERTY> ::= <APARTMENT> | <HOUSE>
<PLACE> ::= <COUNTRY> | <CITY> | <REGION>
```

5.2 Predicative tagging

Application of predicative rules detects in the texts the instances of grammar predicates. Each time a predicate de-

scriptor is found, the process search one of the concepts defined as possible object for this predicate.

Predicates are instantiated until it is impossible to do. This proceeds as follow. Text is processed from left to right. When a predicate's descriptor is recognized, the process looks for a correct object (concept or predicate) for this predicate before the next concept that is a descriptor of another untreated predicate instance. If a correct object is found, the attribute object is given a value for this predicate instance. Next the process tries to give a value to the optional attributes by looking at correct concepts in the text located between the descriptor of this predicate and the next one. After that, the system treats the next descriptor in the text.

If no correct object is found, this descriptor is left and the system immediately treats the next descriptor. This process is made right to the end of text. At this point, if untreated descriptors (that define predicate instance without a found object) are left, the process is repeated from the text's beginning. The operation is repeated until there are no descriptors left to treat or only those that cannot be treated. If such descriptors are left, they are marked as defining empty predicate instances (instances without an object).

Figure 3: Example of predicative tagging

<PURCHASE_1>
<DC_PURCHASE>buy</DC_PURCHASE>
</PURCHASE_1>
<PURCHASE_1 ARG=object>
<PROPERTY>
<APARTMENT>studio</APARTMENT>
<PROPERTY>
</PURCHASE_1 ARG=object>
<PURCHASE_1 ARG=location>
<PLACE><CITY>london</CITY></PLACE>
</PURCHASE_1 ARG=location>
in
</PURCHASE_1 ARG=date>
<DATE>2003</DATE>
</PURCHASE_1 ARG=date>

Therefore we obtain this instance of PURCHASE predicate:

<PURCHASE_1>
[
DESCRIPTOR = buy
OBJECT = studio
DATE = 2003
LOCATION = london
AMOUNT = ∅
]

A predicate P1 can be the object of another one (P2). In this case those of P2 give values to attributes of P1 when possible.

Predicate instantiation is made through a text tagging process. The system tags the descriptor by a predicate reference (the predicate name and an instance number to distinguish different instances of the same predicate). Each predicate attribute is tagged by the predicate reference and its type (Object, Date, Location...).

Example: from the extract described in figure 2, after applying the predicative rules, we obtain the tagging text and the predicate instance described by figure 3.

6 INFORMATION RETRIEVING

After constituent and predicative tagging, tags make concepts and relations clearly readable in the corpus. In the retrieving step, all that has to be done to specify the concepts to be searched. With the tags, the system can easily locate these concepts and their different attributes. In this step empty predicates are ignored. This information feeds a database in which tables correspond to grammar predicate.

7 RESULTS

We have a corpus with around one million records. Each record is taken from an interview between a client and a bank employee. It is composed of a numerical heading and a text area. In the heading, there is an identification number and the recording date. The text area is filled with the interview report written by the employee. The text size varies from record to record: from a few to thirty words. Before text analysis, the text area is treated to make it in conformity with Data Protection Act. Terminological extraction with ANA defines a first set of 15000 term-candidates. After creaming off this set, 1300 remain. Terminological documents (which contain about 350 terms) give 200 new terms. So the terminological step gives us a set of 1500 terms.

7.1 Evaluation method

The goal of this research is to extract client events. These events are client projects and the proposition refusals (from the bank or from the client). The result is a set of searching predicate instances. As there are different attributes for a predicate, three degrees of precision are defined, which depend upon the way these attributes are given a value.

A predicate instance is called valid if the value is correct for attributes, which are given a value (not all the attributes need to be given a value). The validity rate is the number of valid instance per number of instances found.

A valid instance is called totally valid if all of these attributes are given a value and partially valid if one or more attribute is not given a value.

A partially valid instance is called incomplete when one or more attributes is not given a value because of a process mistake and complete when all of the attributes are not given a value because of a lack of information in the corpus.

7.2 Experimentation

Our experiment focuses on a representative sample of 10,000 records taken at random in the corpus. The experiment has been carried out with the aim of extracting the clients' projects from this sample. The results found are validated by experts who have aligned the sample with the table PROJECT in the database filled by our system. According to them, 651 projects are in this sample. The system detects 623 instances of the predicate PROJECT. These instances are detailled as follow : 589 valid instances of which 72 are totally valids. Also the system found 517 partially valid instances of whom 385 are complete and 132 are incomplete.

7.3 Analysis

The coverage rate is not revealing because a lot of records do not contain projects (95%). The recall rate (number of both instances found per number of instances in the corpus) and the validity rate (respectively 95.7% and 94.5%) are both very satisfactory but a lot of instances are partially valid (88% of valid projects). 74.5% of these instances are partially valid because of corpus non-fulfillment but the other 25.5% are imputable to the system. We are working at present to reduce the number of incomplete partially valid instances. For this, after a predicative tagging from left to right, we are considering repeating the process from right to left to first complete optional arguments of predicate instances found and last to try treating the predicate descriptors left by left to right process (defining new predicate instances by detecting objects and arguments). After this right to left step we'll have to repeat the process from left to right because some arguments of new instances can be located on the left of the descriptor.

CONCLUSION

As usual information extraction processes are unusable on poor quality texts, we described a method to extract information from this type of corpus. This approach is based on ontology building led by the type of information to be searched in texts. Good results have been obtained with a very wide cover of information in each record, even for record containing very little information. This method can easily be applied to other corpora and to other domains. The different parts of our system are based on generic methods and do not need modifications to be used with another corpus. Currently experimentations are being carried out to that end.

REFERENCES

[1] Aussenac-Gilles N., Biébow B. and Szulman S., Corpus analysis for conceptual modeling. Proceeding of EKAW'2000, pp. 13-20, Juan-les-Pins, France, 2000.

[2] Aussenac-Gilles N., Bourrigault D., Codamines A. and Gross C., How can knowledge acquisition benefit from terminology. Proceeding of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW '95), Banff, Canada, 1995

[3] Bachimont B., Modélisation linguistique et modélisation logique des ontologies : l'apport de l'ontologie formelle, Proceeding of IC2001, pp. 349-368, Grenoble, France, 2001.

[4] Barry C., Cormier C., Kassel G. and Nobécourt J., Evaluation de langages opérationnels de représentation d'ontologies. Proceeding of IC'2001, pp. 309-327, Grenoble, France, 2001.

[5] Bouaud J., Bachimont B., Charlet J. and Zweigenbaum P., Methodological Principles for Structuring an Ontology. Proceeding of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.

[6] Enguehard C. and Pantéra L., Automatic Natural Acquisition of a Terminology. Journal of quantitative linguistics, Vol. 2, n°1, pp.27-32, 1995.

[7] Guarino N. and Welty W., A Formal Ontology of Properties. Proceedings of the ICAI-00 Work-shop on Applications of Ontologies and Problem-Solving Methods, pp. 12/1-12/8, Las Vegas, United States, 2000.

[8] Guarino N., Concepts, Attributes and Arbitrary Relations : Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. Data & Knowledge Bases Engineering, Vol. 8(2), pp. 249-261, 1992.

[9] Lame G. Knowledge acquisition from texts to-wards an ontology of French law. Proceedings of EKAW'2000, pp. 53-62, Juan-les-Pins, France, 2000.

[10] Nobécourt J., A method to build formal ontologies from texts. Proceedings of EKAW'2000, pp. 21-27, Juan-les-Pins, France, 2000.

[11] Nestorov S. & al., Representative objects : concise representation of semistructured, hierarchical data. Proceedings of International Conference on Data Engineering, pp. 79-90, Birmingham, United Kingdom, 1997.

[12] Swartout B., Patil R., Knight K. and Russ T., Towards distributed use of large-scale ontologies. Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW '96), pp. 32.1-32.19, Banff, Canada, 1996.