Man-Machine Cooperation in Retrieving Knowledge from Technical Texts

Yves Kodratoff[1], Adrian Dimulescu[1] and Ahmed Amrani[1,2]

[1]LRI, Université de Paris-Sud
Bât. 490, 91405 Orsay. France
yk, dadi@lri.fr

[2]ESIEA Recherche
9 rue Vésale, 75005 Paris. France.
amrani@esiea.fr

Abstract

We are presently developing a tool to help field experts to design the series of steps they need in order to be able to recognize the linguistic instances of a set of concepts in texts relative to their field. This involves several challenging steps, from cleaning to concept tagging.

Our approach relies on two basic principles. One is that only a field expert can develop tools able to solve these problems. It follows that the computer scientist should develop user-friendly tools enabling the expert to transfer the expertise to the programs. The second is that inductive tools must also be provided, otherwise the workload is so high that nothing substantial can ever be achieved. The difficulty is that induction has to take place from data that is both incomplete and very noisy; this is a well-known cause of failure in most of the existing inductive programs.

In this paper, we shall describe the way we ask the expert and inductive techniques to cooperate in order to solve three of the crucial steps of knowledge extraction, namely the step of Part-of-Speech tagging, the step of 'terminology' and the step of coreference resolution.

Introduction

The system we are developing tends to favor an AI approach to Knowledge Retrieval, as opposed to the purely statistical one. We thus have to use the knowledge of the domain expert in order to improve the way our system will simulate understanding the content of a text, so as to retrieve significant information from it. We checked this approach in TREC 'Novelty' in 2004 (Amrani et al. 2004), and we are presently competing with this approach in the categorization task of TREC 'Genomics'.

Our system can be divided, more or less arbitrarily, in several steps that actually depend very heavily on each other. In the present paper, and in order to underline its aspects of interactive reasoning, we shall present our system as a linear one, which is almost correct.

Once a corpus has been gathered, the first step is its normalization, also called 'cleaning'. In fact, it includes also the creation of a lexicon, i.e., a list of all the known words, associated with their possible part-of-speech categories. This step is extremely tedious and requires a large number of man-machine interactions that are more of the nature of information exchange than of a shared reasoning.

The second step is the one of Part-of-Speech tagging that will be developed below in section 2. The third step is the one of the locutions, or groups of words, that are significant for a field expert. It will be described in section 3. Finally, section 4 describes briefly our interactive approach to coreference resolution.

Part-of-Speech Tagging

Part-of-Speech Taggers

Part-of-Speech tagging associates to each word with a Morphosyntactic category (Noun, verb, adjective, etc), according to word morphology and context. There are two main approaches for part of speech tagging: the linguistic and the data-driven.

Several data-driven approaches have been applied to Part-of-Speech tagging. Among them, Inductive Logic Programming (Cussens 1997), Transformation-Based Learning (Brill 1994), decision trees learning (Marquez and Rodriguez 1998), support vector machine (Giménez and Marquez 2004) and statistical approaches (Brants 2000) (Cutting et al. 1992) (Toutanova et al. 2003) can be cited. Other sophisticated techniques were used, based on the combination of several kinds of taggers (Brill and Wu 1998) (Halteren, Zavrel, and Daelemans 2001). These techniques are based on the fact that differently designed taggers produce different errors. Thus, the combined tagger shows a higher precision than each isolated ones. Whatever the technology on which they are based, the current taggers obtain a very satisfactory level of precision. The published results are usually about 96-97% of correct tags.

The linguistic approach consists of coding the necessary knowledge in a set of rules written by a linguist. The pioneer system TAGGIT (Greene and Rubin 1971) was used to create the initial tagging of the Brown Corpus, which was then hand revised. One of the most important works of this direction is the development of Constraint Grammars (Karlsson et al. 1995) and their application to POS tagging (Voutilainen 1995), which can be considered the best existing tagger (above of 99% accuracy is reported).

Part-of-Speech Tagging Issues

Although the linguistic approach produces high quality language models, it is a large time-consuming one since many years of human resources are required to develop a good language model.

A prerequisite for the construction of a data-driven PoS-tagger is the availability of a large 'hand-tagged' corpus. Acquiring such a corpus is expensive and time consuming and it is often the bottleneck to build a tagger for a new application or domain.

Most of the available taggers are learned from general corpora. The test and training corpora are of similar nature, and that might explain their high performances. The problem is that the accuracy of general taggers drops down dramatically when applied to new specialized corpora and high quality taggers are not available in specialized domains where a domain expert is needed.

Even when the tagging is very accurate, authors tend to forget to report the nature of the mistakes made by their taggers. For instance, we checked Brill's tagger (Amrani, Kodratoff, and Matte-Tailliez 2004) on a specialized corpus (molecular biology) where it was fairly accurate but some tags reached a high error rate. In fact, some errors are much more harmful than some others. For instance, the word 'that' is very ambiguous, and a mistake on its Part-of-Speech tagging can destroy the structure of the sentence. Similarly, confusions between nouns and verbs entail a large misunderstanding of the sentence. The 'Part-of-Speech tagging community' should develop a register of such harmful errors, and the error rates on these errors should be specified in papers claiming high accuracy.

Some errors occur in doublets or triples that make their correction especially difficult and their destructive role all the more important. In this case, we 'tricked' recursion by using time and side-effects of the transformations. For instance, if we meet the couple *alternate*/VBP *projects*/VBZ (meaning that *alternate* is tagged as a verb and *projects* as a 3rd person singular present verb – and supposing we want to correct it as *alternate*/JJ *projects*/NNS, meaning that an adjective is followed by a plural noun) we introduce a special tag signaling the origin of the corrected tag, say NNS_exVBZ. We then write two rules. The first one says that, in some contexts, if *projects* follows *alternate*/VBP then it has to be re-tagged NNS_exVBZ. The second one says that, in the same context, if *alternate*/VBP is followed by either *projects*/VBZ or *projects*/NNS_exVBZ then it has to be tagged as an adjective. Finally, all tags NNS_exVBZ are transformed back into NNS tags. The actual implementation is a bit more complex to avoid so many particular cases, but it uses this very simple idea.

Part-of-Speech Tagging Methodology: *Progressive Induction*

The correction of difficult Part-of-Speech ambiguities is a significant stage to obtain a 'perfectly' tagged specialized corpus. To correct these ambiguities and to decrease the number of tagging mistakes, we use an approach we call: *Progressive Induction*.

Progressive Induction starts with a general tagger, and corrects gradually the tagging to adapt it to a specialized corpus. The process of correction is iterative. This approach is a combination of machine learning, of rules written by the expert (using a devoted language we developed and called CorTag, described in the next section), and of manual corrections that are iteratively combined in order to obtain an improvement of the tagging while restricting the actions of the expert to the resolution of increasingly delicate problems. The general process is as follows. The expert writes some rules with CorTag so as to decrease the amount of errors relative to a fixed ambiguity. These rules are applied to an initial corpus C_0 and generate an improved corpus $C_{0Expert}$. A rule learning algorithm is used to generate a model of the change from corpus C_0 to $C_{0Expert}$. These rules are applied to C_0 and generate a third corpus, C_{0Ind}. The differences between $C_{0Expert}$ and C_{0Ind} are then analyzed by the expert, with the help of ETIQ. These differences help to

- point at mistakes made by the expert.
- improve the rules of the expert. The rules that 'won over' the expert are shown and help in this improvement process.
- when corrected on the fly by the expert, critical tags are made correct and the corpus itself is improved for a better future use.

Below we will describe in detail the CorTag language and our approach to progressive induction.

A Tagging Language: CorTag. All this explain why we developed a language to correct the tags in the texts. Since we suppose that we have already a lexicon and tagged texts (both certainly with mistakes) this language deals essentially with the relational tagging problems. It is in the form of rules handling triplets describing, in this order, the place of the words in the sentence relative to a fixed word of interest placed at '0', the word itself, and its tag. For instance, in order to avoid writing overly specific rules, we would build up *list1* containing 'alternate' and *list2* containing 'projects', and the rules described earlier would be written:

if [context] (0,@list1,VBP) (+1,@list2,VBZ) then (+1, ,NNS_exVBZ)

if [context] (0, @list1,VBP) (+1, @list2,or(VBZ,NNS_exVBZ)) then (0, ,JJ) (+1,,NNS)

This language will be put in free access on the Internet as soon as its user's guide is ready. We thus shall not give too many details about it here. The rules we gave above describe the simplest cases. In general, it is designed to write, using relatively simple formula, quite complex relationships among words of one sentence. For instance, for tagging properly a 'that' as an 'IN' (Preposition or subordinating conjunction) , we wrote a rule stating:

"tag as 'IN' a 'that' when the first verb before this 'that' belongs to a given list, except when there is a singular or plural noun (not a name) between 'that' and this verb."

A very important feature of the language is that it easily describes the cases where optional words exist inside a relation. For instance, the premise of a rule describing a determiner followed by a noun is written as

(-1,,DT) (0,,NN)

If between the noun and the determiner, one adverb (called 'RB') and two adjectives with coordination between them are possible but not necessary, we simply write this premise as:

(?,,DT) (*1,,or(JJ,RB)) (*1,,JJ) (*1,,CC) (*1,,JJ) (0,,NN).

Tagging by Progressive Induction. The initial stage of progressive induction (Stage 0, corpus C_0) consists in the obtainment of a corpus of specialty tagged by a Part-of-Speech tagger trained on a general corpus (which is of different nature of our corpus).

The expert then identifies the confusions that seem to be the most significant ones, and writes correction rules for each confusion. These rules can be written within ETIQ (Amrani, Kodratoff, and Matte-Tailliez 2004) or, if they are strongly contextual rules, by using the language CorTag, which is dedicated to the drafting of these rules. A fast description of this language is given above. Each rule applies to a precise context and is used to fix a specific confusion. The expert thus produces a certain number of rules, which are applied to the corpus (stage 1, producing $C_{0Expert}$) and he thus checks the validity of his rules. A quasi insolvable problem arises when the expert works on a large corpus: the number of application of the rules can be of several thousands, thus the expert can check the validity of his rules only on a subcorpus of the initial one.

A learning base is generated from $C_{0Expert}$ and C_0. This base contains the examples modified by the rules of the expert (positive examples) and the non-modified examples (negative examples). Applying the induced rules on the corpus at its 0 stage, we can generate a third version, corpus C_{0Ind}, the one of stage 2.

Lastly, the corpora of stages 1 and 2 are presented at the expert when the corrections made by the rules of the expert differ from those brought by the induced rules. The expert examines these cases. A significant point is to notice that the expert must either confirm 'his' tagging, or that the induced rules are right to contradict his rules. When this case happens enough frequently (i.e. the inductive process is of good quality), this produces a kind of competition between the expert and the induction, so that his attention stays constantly excited.

Thus we have three successive versions of the same corpus: $ExpRulCorp_0$, $IndRulCorp_0$ and $SureCorp_0$.

- $ExpRulCorp_0$ is the result of applying the expert rules to the starting corpus.
- $IndRulCorp_0$ is the result of applying the induced rules to the starting corpus.
- $SureCorp_0$ is the corpus in which we keep the labels manually fixed (if the induction 'won') or confirmed by the expert (if he 'won') during the process.

Once again, if the corpus is sufficiently small, the expert can fix all the errors and it is not really necessary to reiterate this process. As we work on the assumption that the corpus is bulky, the expert cannot examine the thousands of cases where there is a difference between his rules and the induced ones. On the other hand, he can notice some of the cases where he 'lost' against induction and he can then use ETIQ to display the 'winning' induced rule. This rule can be analyzed to provide some indication on the way of improving the existing rules.

It then applies these new rules to $SureCorp_0$ which becomes the starting corpus of the next iteration, so as to generate $ExpRulCorp_1$ by applying to it the rules deduced during the preceding iteration.

It is of course theoretically possible that the expert is completely mistaken and that iteration 'n+1' contains more errors than iteration 'n', and that would be a case of failure of our method, pushing the expert to quit. In fact, being shown his errors, the expert tends rather to correct them and to obtain more effective rules. The danger would be that the system induces rules making the same errors as with the preceding iteration, and than the expert re-examines these same errors at each iteration (and that would make him quit by being bored). We never noted this behavior because our inductive system learns very different rules when the training sets are different. Note in the passing the importance of the manual corrections of the expert, even if they are relatively few, in order to start with a really different tagging.

Consequently, and in practice, the expert will notice a decrease, at each iteration, of the number of times when the inductive system wins. As a consequence, after some iterations he can examine all the differences and, if required, he can fix all his errors manually, without resorting to rules. The final corpus, $SureCorp_{0p}$ is then perfectly tagged from the point of view of the tagging error considered at the beginning.

It is obviously necessary to repeat the whole process for each tagging error that needs to be fixed.

The most significant mistakes are those which prevent the comprehension of the sentence by destroying its syntactic structure. The ones we corrected on the two corpora on which we worked (the one of TREC Novelty of approximately 9 megabytes and the training one of TREC Genomics of approximately 32 megabytes) are as follows:

- tagging mistakes of 'that' which can be conjunction, relative, determinant or pronoun
- confusion of the -ed and –ing forms in their verbal use and their use as premodifiers (and which we then label as adjectives)
- confusion of the nouns and the verbs, including the special case of the 3rd person of the present singular
- with less importance, confusion between past participles and verb in the past tense.

Thus, the process must be entirely repeated 3 or 4 times, which is not too much in view of the advantage to have a correctly tagged corpus.

Example: Let us consider the confusion between the tags NNS (Noun, plural) and VBZ (Verb, 3rd person singular

present). By examining, C_0, the expert imagines some rules that improve the tagging situation over the automatic tagger. Using these rules, the rule improved corpus, $ExpRulCorp_0$, is obtained. Using ETIQ, we display the tags VBZ and NNS that are different in these two corpora and, as explained above, we learn rules based on this difference. In the experiment we did to generate the present example, it happened that 50 rules were generated. Among them, let us examine the rule:
"IF the word under study is already tagged by NNS or by VBZ *AND* it is followed by a determinant (DT) *THEN* tag it VBZ" (that is, a VBZ is often followed by a DT).
Relative to $ExpRulCorp_0$, this rule applies correctly 630 times (in the corpus we used) and incorrectly 20 times. We display the 630 sentences where a VBZ is followed by a DT and realize that these 630 tags are correct. This rule must thus contain some linguistic truth. We then display the 20 sentences where a NNS is followed by a DT. We observe five cases.
Case 1: the DT is 'both' or 'each'. Therefore, the rule was a bit over general and should read "… followed by a DT, except 'each' or 'both'". This hints at the fact that all DT are not equivalent, and we create classes of DT that solve this problem. These classes may be (and actually were) useful in order to solve other kinds of overgeneralizations.
Case 2: a coma that should appear after the NNS, but it has been omitted. We make a list of the nouns plural after which the authors tend to forget the coma, and add this list in the exceptions to the above rule.
Case 3: the 'DT' is in fact an 'A' (as in "proteins A and B") and the 'A' was incorrectly tagged as a DT. We have to correct this mistake anyhow by applying a rule that corrects this wrong tagging.
Case 4: the word of interest was incorrectly tagged as an NNS, this mistake should also be corrected.
Case 5: the word of interest was incorrectly tagged as an NNS by the expert rules, and should be a VBZ. as expected.

This very simple example illustrates the power of combining human and machine inductive abilities, especially in cases were the mistakes cannot be corrected all at once but ask for corrections that are embedded. A complex recursive program would be needed to solve automatically these cases, while the human reactions cut down the complexity.

Once an 'almost perfectly' tagged subcorpus has been obtained on a significant part of the corpus (we evaluate it at some 1/10 of the corpus), powerful statistical learning techniques, such as the Hidden Markov Chains can be applied with success. We noticed, however, that the above described 4 difficult cases are badly handled by these techniques (most humans have problems handling them properly!) and we need to clean the statistically tagged corpus by using rules specific to these hard cases. In other words, the collaboration between expert and machines never stops being useful.

Similar Approaches

Let us now discuss two similar approaches. ANNOTATE is an interactive tool for semi-automatic tagging (Plaehn and Brants 2000) similar to ours. This system interacts with a statistical tagger (Brants 2000) and a "parser" to accelerate the tagging. Grammatical tagging works as follows: the tagger determines the tag of each word on the basis of the tag frequency probability, the system then displays the sentences and calls for confirmation or correction of the non reliable labels. In this system, the annotator does not write rules. The internal model is updated gradually with the confirmations and the corrections carried out. These changes are thus exploited immediately.

KCAT (Won-Ho et al. 2000) is a semi-automatic annotation tool to build a precise and consistent tagged corpus. It combines the use of a statistical tagger, and lexical rules of clarification acquired by an expert. The rules used are limited to word similarity. The method enables correcting the words with non-reliable labels. The expert does not have to repeat the same action on two words in the same context because lexical rules are generated automatically. The expert tags words by lexical rules, the remaining words are tagged by the statistical tagger, then the expert has to correct directly the unreliable tags. These rules are very specific. In order to obtain a well-tagged corpus, the expert must insert a large quantity of tags. This increases human costs and slows down the speed of execution.

Terminology

In order to find all the 'interesting' collocations in a technical text, we developed a method which is finally quite complex but relies on a large amount of information exchange between the expert and a system that learns interactively to detect significant collocations.

Determining the Syntactic Properties of an 'Interesting' Collocation

The expert is requested to find ways, typical of his field, to underline important sequences of words. For instance, in biology, a sentence of the form :
'the function of' + 'adjectives and nouns' + 'verb'
underlines the idea that the sequence of 'adjectives and nouns' constitutes a significant collocation of them. They are obviously many such typical forms, and the expert must provide some of them. These syntactic properties are then expressed in our language CorTag in order to gather all the collocations in this position. In the case, of TREC Genomics, for example, since the corpus amounts to some 450 megabytes of usable text, the total number of such collocations is very large.

The inverse approach is then followed : knowing a first set of such significant collocations, find in which syntactic environment they occur.

These two methods alternate under the field expert's control until no new collocations are detected.

This enables us to set up a list of words that tend to generate interesting collocations, such as 'gene', 'site' etc. The expert could not directly provide such a list. For instance, we presently use a list of some 15 000 such words.

Using an Automatic Term Generator

As we underline in (Roche et al. 2003), many statistical methods have been used in order to take into account the degree of surprise or coincidence of collocations. The user of our system can chose one of the available methods, but it is always weighted by the presence of the words in the list of words found in the already built list of interesting collocations.

Besides, we developed an interactive implementation in which the user can directly accept or reject a collocation.

In this case, there is no inductive system, but a statistical system, the parameters of which are interactively modified by a user who inputs what field knowledge shows to be important. As in the case of Part-of-Speech tagging, a completely manual approach is totally impossible. For instance, on the training set of TREC Genomics, we already spotted some 300 000 important terms, many of them occurring only once within the 32 megabytes of this training set.

Coreference resolution

Instead of the more classical pronominal anaphora, texts in Genomics tend to use sortal anaphora, as for example, *"these genes"*, *"both proteins"*, *"the promoter"*, that make reference to their antecedents using their semantic type. They are obviously loaded with semantics, and we decided to use an interactive approach in order to build the ontologies necessary to the solution of this type of coreferences.

Interactive building of the leaves of the ontology

With expert help, we decide of a few 'canonical' ways of expressing biological properties. The preceding tagging phase enables us to define contextual expression depending on words, and/or on the tag of a word, giving much more generality to the patterns to be recognized. For instance, the name of a protein is often given in between a determinant and the words 'protein' or 'proteins' as in *"the p37 and p40 proteins"*. A systematic survey of similar syntaxic forms over several hundreds of mega-bytes provides an impressive list of protein names. Importantly, these names are not as they are given in the classical biological repositories, but as they actually occur in the literature. Interactions are performed in order to increase the amount of our so-called canonical ways of writing, this yielding more instances of the concepts. By using our language CorTag, the field expert can develop and check the value of the proposed canonical ways of writing.

Interactive building of the structure of the ontology

In a well-known field, such as the organization of a company, an expert can provide the complete ontology structure, that is the ontology minus the instances of its concepts. In Biology, GeneOntology is a good example of a consistent attempt in this direction. However, building a complete ontology seems to be "a biologist's dream", as one of our expert stated, since it is so complex, since it evolves constantly, and since it has to be partly topic specific. The same approach as the one described just above proved to be very successful in providing the expert with a incomplete structure that can be modified. For instance, lexical constructs like: *"the stress inducible proteins hsp60 and GRP94"* or *"CML is a clonal disorder"* provide the following structuring information: *"stress inducible protein"* is a subclass of *"protein"* and *"clonal disorder"* is a subclass of *"disorder"*. Actually, the typical form of an explicit generalization construct is "*A* is a *B* that *C*", and besides the generalization relation we can also infer the property *C* of the concept *A*. An example of such a case is: "*necdin* ***is a*** *unique growth suppressor* ***that*** *blocks cell_cycle reentry* ***and*** *promotes survival of postmitotic neurons"*.

The biology expert reacts to these proposals by asking specific information of interest and by proposing lexical patterns for its detection. In fact, we also use lists of verbs of biological importance that can mark the links in the ontology. For instance, the above sentence provides links of the nature "*block*" and "*promote*."

Learning to resolve sortal anaphora

We built a scoring measure in order to sort the words or collocations being possible antecedents to a coreference, and we used version 2.0 of WordNet in order to check WordNet permitted structures in the ontology. After this is applied, the expert is asked to review the failures and to modify the structure of the ontology in order to resolve these failure cases. For instance, when applying our method to the sentences : *"...chronic myelogenous leukemia is a myeloproliferative disorder that, over time, progresses to acute leukemia. Both processes..."* we tried to resolve *both processes* by inferring that *leukemia* is a *disorder* and *disorder* is a *process*. By exploring the ontology support we find knowledge that *leukemia* is a *disorder* but, as of version 2.0 of WordNet, a *disorder* does not appear as being a *process*. The field expert then is urged not only to specify that a disorder is a process, but also to make precise which disorders a looked upon as processes in Molecular Biology and which are not. The ontology thus obtained is specific to Molecular Biology and should not be applied, for instance, to airplane maintenance. Another ontology would have to be built, starting from texts relative to, say, airplane maintenance.

Conclusion

The simulation of the understanding of technical texts is certainly a vastly complex problem that requires to be dealt with by a system of high complexity. Depending on the problem at hand, various inductive or deductive techniques can solve this problem for a very small corpus. In order to deal with the extra complexity due to the large amount of texts to be processed, the field expert must be included inside each reasoning step. This cooperation can take various forms, again depending on the problem to be solved.

References

Amrani, A., Azé, J., Heitz, T., Kodratoff, Y., and Roche, M. 2004. From the texts to the concepts they contain: a chain of linguistic treatments. In Proceedings of TREC'04 (Text REtrieval Conference), Gaithersburg Maryland.

Amrani, A., Kodratoff Y., and Matte-Tailliez O. 2004. A Semi-automatic System for Tagging Specialized Corpora. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004), 670-681.

Brants, T. 2000. TnT - A Statistical Part-of-Speech Tagger, In Proceedings of the 6th Conference on Applied Natural Language Processing, Seattle.

Brill, E. 1994. Some Advances in Transformation-Based Part of Speech Tagging, AAAI, Vol. 1, 722–727.

Brill, E., and Wu, J. 1998. Classifier Combination for Improved Lexical Disambiguation. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics.

Cussens, J. 1997. Part-of-speech tagging using Progol. S. Dzeroski et N. Lavrac, Eds. In Proceedings of the 7th International Workshop on ILP, Vol.1297(1997),93–108.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. 1992. A practicial part-of-speech tagger, In Proceedings of the 3rd Conference on Applied Natural Language Processing.

Giménez, J., and Marquez, L. 2004. Fast and accurate part-of-speech tagging: The SVM approach revisited. Recent Advances in Natural Language Processing, 153-163.

Greene, K., and Rubin, D. 1971. Automated Grammatical Tagging of English. Department of Linguistics, Brown University, Providence, Rhode Island.

Halteren, V., Zavrel, J., and Daelemans, W. 2001. Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems, Computational linguistics Vol. 27 (2001), 199–229

Karlsson, F., Voutilainen, A., Heikkila, J., and Anttila, A. (editors). Constraint Grammar: A Language–Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin and New York, 1995.

Marquez, L., and Rodriguez, H. 1998. Part-of-Speech Tagging Using Decision Trees, In Proceedings of ECML 1998, 25–36.

Plaehn, O., and Brants, T. 2000. Annotate - An Efficient Interactive Annotation Tool. In Proceedings of the Sixth Conference on Applied Natural Language Processing. Seattle (2000).

Roche, M., Matte-Tailliez, O. , Azé, J., and Kodratoff, Y. 2003. Extraction de la Terminologie du Domaine : Etude de Mesures sur un Corpus Spécialisé Issu du Web. Actes de la conférence les Journées Francophones de la Toile 2003, 279-288, Tours, France.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, 252-259.

Voutilainen, A. 1995. A syntax-based part-of-speech analyser. In Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics, Dublin, 1995.

Won-Ho, R., Heui-Seok, L., Jin-Dong, K., and Hae-Chang R. 2000. KCAT: A Korean Corpus Annotating Tool Minimizing Human Intervention. In Proceedings of the Eighteenth International Conference on Computational Linguistics, 1096-1100.