Answer Extraction
Towards better Evaluations of NLP Systems

Rolf Schwitter and **Diego Mollá** and **Rachel Fournier** and **Michael Hess**
Department of Information Technology
Computational Linguistics Group
University of Zurich
CH-8057 Zurich
{schwitter, molla, fournier, hess}@ifi.unizh.ch

Abstract

We argue that reading comprehension tests are not particularly suited for the evaluation of NLP systems. Reading comprehension tests are specifically designed to evaluate human reading skills, and these require vast amounts of world knowledge and common-sense reasoning capabilities. Experience has shown that this kind of full-fledged question answering (QA) over texts from a wide range of domains is so difficult for machines as to be far beyond the present state of the art of NLP. To advance the field we propose a much more modest evaluation set-up, viz. Answer Extraction (AE) over texts from highly restricted domains. AE aims at retrieving those sentences from documents that contain the explicit answer to a user query. AE is less ambitious than full-fledged QA but has a number of important advantages over QA. It relies mainly on linguistic knowledge and needs only a very limited amount of world knowledge and few inference rules. However, it requires the solution of a number of key linguistic problems. This makes AE a suitable task to advance NLP techniques in a measurable way. Finally, there is a real demand for working AE systems in technical domains. We outline how evaluation procedures for AE systems over real world domains might look like and discuss their feasibility.

1 On the Design of Evaluation Methods for NLP Systems

The idea that the systematic and principled evaluation of document processing systems is crucial for the development of the field as a whole has gained wide acceptance in the community during the last decade. In a number of large-scale projects (among them TREC (Voorhees and Harman, 1998) and MUC (MUC-7, 1998)), evaluation procedures for specific types of systems have been used extensively, and refined over the years. Three things were common to these evaluations: First, the systems to be evaluated were each very closely tied to a particular task (document retrieval and information extraction, respectively). Second, the evaluation was of the black box type, i.e. it considered only system input-output relations without regard to the specific mechanisms by which the outputs were obtained. Third, the amount of data to be processed was enormous (several gigabytes for TREC).

There is general agreement that these competitive evaluations had a striking and beneficial effect on the performance of the various systems tested over the years. However, it is also recognized (albeit less generally) that these evaluation experiments also had the, less beneficial, effect that the participating systems focussed increasingly more narrowly on those few parameters that were measured in the evaluation, to the detriment of more general properties. In some cases this meant that powerful and linguistically interesting but slow systems were dropped in favour of shallow but fast systems with precious little linguistic content. Thus the system with which SRI participated in the MUC-3 evaluation in 1991, TACITUS (Hobbs et al., 1991), a true text-understanding system, was later replaced by FASTUS (Appelt et al., 1995; Hobbs et al., 1996), a much simpler, and vastly faster, information extraction system. The reason was that TACITUS was spending so much of its time attempting to make sense of portions of the text that were irrelevant to the task that recall was mediocre. We argue that the set-up of these competitive evaluations, and in particular the three parameters mentioned above, drove the development of the participating systems towards becoming impressive feats of engineering, fine-tuned to one very specific task, but with limited relevance outside this task and with little linguistically relevant content. We argue that these evaluations therefore did not drive progress in Computational Linguistics very much.

We therefore think it a timely idea to conceive of evaluation methodologies which measure the *linguistically* relevant functions of NLP systems and thus advance Computational Linguistics as a science rather than as an engineering discipline. The suggestion made by the organizers of this workshop on how this could be achieved has four components. First, they suggest to use full-fledged text-based question answering (QA) as *task*. Second, they suggest a relatively *small amount of text* (compared with the volumes of text used in TREC) as test data. Third they (seem to) suggest to use texts from a *wide range of domains*. Finally they suggest to use pre-existing question/answer pairs, developed for and tested on humans, as *evaluation benchmark* (Hirschman et al., 1999).

However, our experience in the field leads us to believe that this evaluation set-up will not help Computational Linguistics as much as it would be needed, mainly because it is way too ambitious. We fear that this fact will force developers, again, to design all kinds of ad-hoc solutions and efficiency hacks which will severely limit the scientific relevance of the resulting systems. We argue that three of the four components of the suggested set-up must be reduced considerably in scope to make the test-bed helpful.

First, we think the *task* is too difficult. Full-fledged QA on the basis of natural language texts is far beyond the present state of the art. The example of the text-based QA system LILOG (Herzog and Rollinger, 1991) has shown that the analysis of texts to the depth required for real QA over their contents is so resource intensive as to be unaffordable in any real world context. After an investment of around 65 person-years of work the LILOG system could answer questions over a few (reputedly merely three) texts of around one page length each from an extremely narrow domain (city guides and the like). We think it is fair to say that the situation in our field has not changed enough in the meantime to invalidate this finding.

Second, we agree that the *volume of data* to be used should be relatively small. We must avoid that the sheer pressure of the volumes of texts to be processed forces system developers to use shallow methods.

Third, we think it is very important to restrict the *domain* of the task. We certainly do not argue in favour of some toy domain but we get the impression that the reading comprehension texts under consideration cover a far too wide range of topics. We think that technical manuals are a better choice. They cover a narrow domain (such as computer operating systems, or airplanes), and they also use a relatively restricted type of language with a reasonably clear semantic foundation.

Fourth, we think that tests that are specifically designed to evaluate to what extent a *human* being understands a text are intrinsically unsuitable for our present purposes. Although it would admittedly be very convenient to have "well written" texts, "good" questions about them and the "correct" answers all in one package, the texts are not "real world" language (in that they were written specifically for these tests), and the questions are just far too difficult, primarily because they rely on exactly those components of language understanding where humans excel and computers are abominably poor (inferences over world knowledge).

In Section 2 we outline what kinds of problems would have to be solved by a QA system if it were to answer the test questions given in (WRC, 2000). Most of the problems would require enormous amounts of world knowledge and vast numbers of lexical inference rules for a solution, on top of all the "classical" linguistic problems our field has been struggling with (ambiguities, anaphoric references, synonymy/hyponymy). We will then argue in Section 3 that a more restricted kind of task, Answer Extraction, is better suited as experimental set-up as it would focus our forces on these unsolved but reasonably well-understood problems, rather than divert them to the ill-understood and fathomless black hole of world knowledge. In Section 4, we will finally outline how evaluation procedures in this context might look like.

2 Why Reading Comprehension Tests via QA are Too Difficult

Reading comprehension tests are designed to measure how well human readers understand what they read. Each story comes with a set of questions about information that is stated or implied in the text. The readers demonstrate their understanding of the story by answering the questions about it. Thus, reading comprehension tests assume a cognitive process of human beings. This process involves expanding the mental model of a text by using its implications and presuppositions, retrieving the stored information, performing inferences to make implicit information explicit, and generating the surface strings that express this information. Many different forms of knowledge take part in this process: linguistic, procedural and world knowledge. All these forms coalesce in the memory of the reader and it is very difficult to clearly distinguish and reconstruct them in a QA system. At first sight the story published in (WRC, 2000) is easy to understand because the sentences are short and cohesive. But it turns out that a classic QA system would need vast amounts of knowledge and inference rules in order to understand the text and to give sensible answers.

Let us investigate what kind of information a full-fledged QA system needs in order to answer the questions that come with the reading comprehension test (Figure 1) and discuss how difficult it is to provide this information.

To answer the first question

(1) *Who collects maple sap?*

the system needs to know that the mass noun *sap* in the text sentence

Farmers collect the sap.

is indeed the *maple sap* mentioned in the question. The compound noun *maple sap* is a semantically narrower term than the noun *sap* and encodes an implicit relation between the first element *maple* and the head noun *sap*. This relation names the origin of the material. Since no explicit information about the relation between the two objects is available in the text an ideal QA system would have to assume such a relation by a form of abductive reasoning.

How Maple Syrup is Made

Maple syrup comes from sugar maple trees. At one time, maple syrup was used to make sugar. This is why the tree is called a “sugar” maple tree.
Sugar maple trees make sap. Farmers collect the sap. The best time to collect sap is in February and March. The nights must be cold and the days warm.
The framer drills a few small holes in each tree. He puts a spout in each hole. Then he hangs a bucket on the end of each spout. The bucket has a cover to keep rain and snow out. The sap drips into the bucket. About 10 gallons of sap come from each hole.

1. Who collects maple sap?
   (Farmers)
2. What does the farmer hang from a spout?
   (A bucket)
3. When is sap collected?
   (February and March)
4. Where does the maple sap come from?
   (Sugar maple trees)
5. Why is the bucket covered?
   (to keep rain and snow out)

Figure 1: Reading comprehension test

To answer the second question

(2) *What does the farmer hang from a spout?*

successfully the system would need at least three different kinds of knowledge:

First, it would need discourse knowledge to resolve the intersentential co-reference between the anaphor *he* and the antecedent *the farmer* in the following text sequence:

The farmer drills a few small holes in each tree. [...] Then he hangs a bucket ...

Although locating antecedents has proved to be one of the hard problems of natural language processing, the anaphoric reference resolution can be done easily in this case because the antecedent is the most recent preceding noun phrase that agrees in gender, number and person.

Second, the system would require linguistic knowledge to deal with the synonymy relation between *hang on* and *hang from*, and the attachment ambiguity of the prepositional phrase used in the text sentence and the query.

Third, the system needs an inference rule that makes somehow clear that the noun phrase *a spout* expressed in the query is entailed in the more complex noun phrase *the end of each spout* in the text sentence. Additionally, to process this relation the system would require an inference rule of the form:

> **IF** X does Y to **EACH** Z
> **THEN** X does Y to **A** Z.

The third question

(3) *When is sap collected?*

asks for the time point when sap is collected but the text gives only a rule-like recommendation

> *The best time to collect sap is in February and March.*

with an additional constraint

> *The nights must be cold and the days warm.*

and does not say that the sap is in fact collected in February and March. The bridging inference that the system would need to model here is not founded on linguistic knowledge but on world knowledge. Solving this problem is very hard. It could be argued that default rules may solve such problems but it is not clear whether formal methods are able to handle the sort of default reasoning required for representing common-sense reasoning.

To give an answer for the fourth question

(4) *Where does the maple sap come from?*

the system needs to know that maple sap comes from sugar maple trees. This information is not explicitly available in the text. Instead of saying where *maple sap* comes from the text says where *maple syrup* comes from:

> *Maple syrup comes from sugar maple trees.*

There exists a metonymy relation between these two compound nouns. The compound noun *maple syrup* (i.e. product) can only be substituted by *maple sap* (i.e. material), if the system is able to deal with metonymy. Together with the information in the sentence

> *Sugar maple trees make sap.*

and an additional lexical inference rule in form of a meaning postulate

> **IF** X makes Y **THEN** Y comes from X.

the system could deduce (in theory) first *sap* and then by abductive reasoning assume that the *sap* found is *maple sap*. Meaning postulates are true by virtue of the meaning they link. Observation cannot prove them false.

To answer the fifth question

(5) *Why is the bucket covered?*

the system needs to know that the syntactically different expressions *has a cover* and *is covered* have the same propositional content. The system needs an explicit lexical inference rule in form of a conditional equivalence

> **IF** Conditions
> **THEN** X has a cover <=> X is covered.

that converts the verbal phrase with the nominal expression into a the corresponding passive construction (and vice versa) taking the present context into consideration.

As these concrete examples show, the task of QA over this simple piece of text is frighteningly difficult. Finding the correct answers to the questions requires far more information that one would think at first. Apart from linguistic knowledge a vast amount of world knowledge and a number of bridging inferences are necessary to answer these seemingly simple questions. For human beings bridging inferences are automatic and for the most part unconscious. The hard task consists in reconstructing all this information coming from different knowledge sources and modeling the suitable inference rules in a general way so that the system scales up.

3 Answer Extraction as an Alternative Task

An alternative to QA is answer extraction (AE). The *general* goal of AE is the same as that of QA, to find answers to user queries in textual documents. But the way to achieve this is different. Instead of generating the answer from the information given in the text (possibly in implicit form only), an AE system will retrieve the specific sentence(s) in the text that contain(s) the explicit answer to the query. In addition, those phrases in the sentence that represent the explicit answer to the query may be highlighted. For example, let us assume that the following sentence is in the text (and we are going to use examples from a technical domain, that of the Unix user's manual):

(1) *cp copies the contents of filename1 onto filename2.*

If the user asks the query

Which command copies files?

a QA system will return:

cp

However, an AE system will return all the sentences in the text that directly answer the question, among them (1).

Obviously, an AE system is far less powerful than a real QA system. Information that is not explicit in a text will not be found, let alone information that must be derived from textual information together with world knowledge. But AE has a number of important advantages over QA as a test paradigm. First, an obvious advantage of this approach is that the user receives first-hand information, right from the text, rather than system-generated replies. It is therefore much easier for the user to determine whether the *result* is *reliable*. Second, it is a *realistic task* (as the systems we are describing below proves) as there is no need to generate natural language output, and there is less need to perform complex inferences because it merely looks up things in the texts which are explicitly there. It need not use world knowledge. Third, it requires the solution of a number of well-defined and truly *important linguistic problems* and is therefore well suited to measure, and advance, progress in these respects. We will come to this later. And finally, there is a real *demand for working AE systems* in technical domains since the standard IR approaches just do not work in a satisfactory manner in many applications where the user is in pressure to quickly find a specific answer to a specific question, and not just (potentially long) lists of pointers to (potentially large) documents that may (or may not) be relevant to the query. Examples of applications are on-line software help systems, interfaces to machine-readable technical manuals, help desk systems in large organizations, and public enquiry systems accessible over the Web.

The basic procedure we use in our approach to AE is as follows: In an off-line stage, the documents are processed and the core meaning of each sentence is extracted and stored as so-called minimal logical forms. In an on-line stage, the user query is also processed to produce a minimal logical form. In order to retrieve answer sentences from the document collection, the minimal logical form of the query is proved, by a theorem prover, over the minimal logical forms of the entire document collection (Mollá et al., 1998). Note that this method will *not* retrieve patently wrong answer sentences like *bkup files all copies on the hard disk* in response to queries like *Which command copies files?* This is the kind of response we inevitably get if we use some variation of the bag-of-words approach adopted by IR based systems not performing any kind of content analysis.

We are currently developing two AE systems. The first, ExtrAns, uses deep linguistic analysis to perform AE over the Unix manpages. The prototype of this system uses 500 Unix manpages, and it can be tested over the Web [http://www.ifi.unizh.ch/cl/extrans]. In the second (new) project, WebExtrAns, we intend to perform AE over the "Aircraft Maintenance Manual" of the Airbus 320 (ADRES, 1996). The larger volume of data (about 900 kg of printed paper!) will represent an opportunity to test the scalability of an AE system that uses deep linguistic analysis.

There is a number of important areas of research that ExtrAns and WebExtrAns, and by extension any AE system, has to focus on. First of all, in order to generate the logical form of the sentences, the following must be tackled: Finding the verb arguments, performing disambiguation, anaphora resolution, and coping with nominalizations, passives, ditransitives, compound nouns, synonymy, and hyponymy (Mollá et al., 1998; Mollá and Hess, 2000). Second, the very idea of producing the logical forms of real-world text requires the formalization of the logical form notation so that it is expressive enough but still remaining usable (Schwitter et al., 1999). Finally, the goal of producing a practical system for a real-world application needs to address the issue of robustness and scalability (Mollá and Hess, 1999).

Note that the fact that AE and QA share the same goal makes it possible to start a project that initially performs AE, and gradually enhance and extend it with inference and generation modules, until we get a full-fledged QA system. This is the long-time goal of our current series of projects on AE.

4 Evaluating the Results

Instead of using reading comprehension tests that are meant for humans, not machines, we should produce the specific tests that would evaluate the AE capability of machines. Here is our proposal.

Concerning *test queries*, it is always better to use real world queries than queries that were artificially constructed to match a portion of text. Experience has shown time and again that real people tend to come up with questions different from those the test designers could think of. By using, as we suggest, manuals of real world systems, it is possible to tap the interaction of real users with this system as a source of real questions (we do this by logging the questions submitted to our system over the Web). Another way of finding queries is to consult the FAQ lists concerning a given system sometimes available on the Web. In both cases you will have to filter out those queries that have no answers in the document collection or that are clearly beyond the scope of the system to evaluate (for example, if the inference needed to answer a query is too complex, even for a human judge).

Concerning *answers*, the principal measures for the AE task must be recall and precision, applied to individual answer sentences. *Recall* is the number of correct answer sentences the system retrieved divided by the total number of correct answers in the entire document collection. *Precision* is the number of correct answer sentences the system retrieved divided by the total number of answers it returned. As is known all too well, recall is nearly impossible to determine in an exact fashion for all but toy applications since the totality of correct answers in the entire document collection has to be found mainly by hand. Almost certainly one will have to resort to (hopefully) representative samples of documents to arrive at a reasonable approximation to this value. Precision is easier to determine although even this step can become very time consuming in real world applications.

If, on the other hand, one only needs to do an approximate evaluation of the AE system, it would be possible to find a representative set of correct answers by making a person write the ideal answers, and then automatically finding the sentences in the documents that are semantically close to these ideal answers. Semantic closeness between a sentence and the ideal answer can be computed by combining the *succinctness* and *correctness* of the sentence with respect to the ideal answer. Succinctness and correctness are the counterparts of precision and recall, but on the sentence level. These measures can be computed by checking the overlap of words between the sentence and the ideal answer (Hirschman et al., 1999), but we suggest a more content-based approach.

Our proposal is to compare not words in a sentence, but their logical forms. Of course, this comparison can be done only if it is possible to agree on how logical forms should look like, to compute them, and to perform comparisons between them. The second and third conditions can be fulfilled if the logical forms are simple lists of predicates that contain some minimal semantic information, as it is the case in ExtrAns (Schwitter et al., 1999). In this paper we will use a simplification of the minimal logical forms used by ExtrAns. Below are two sentences with their logical forms:

(1) *rm removes one or more files.*
remove(x,y), rm(x), file(y)

(2) *csplit prints the character counts for each file created, and removes any files it creates if an error occurs.*

print(x,y), csplit(x), character-count(y), **remove(x,z)**, **file(z)**, create(x,z), occur(e), error(e)

As an example of how to compute succinctness and correctness, take the following question:

Which command removes files?

The ideal answer is a full sentence that contains the information given by the question and the information requested. Since *rm* is the command used to remove files, the ideal answer is:

rm removes files.
remove(x,y), rm(x), file(y)

Instead of computing the overlap of words, succinctness and correctness of a sentence can be determined by computing the overlap of predicates. The overlap of the predicates (overlap henceforth) of two sentences is the maximum set of predicates that can be used as part of the logical form in both sentences. The predicates in boldface in the two examples above indicate the overlap with the ideal answer: 3 for (1), and 2 for (2).

Succinctness of a sentence with respect to an ideal answer (precision on the sentence level) is the ratio between the overlap and the total number of predicates in the sentence. Succinctness is, therefore, 3/3=1 for (1), and 2/8=0.25 for (2).

Correctness of a sentence with respect to an ideal answer (recall on the sentence level) is the ratio between the overlap and the number of predicates in the ideal answer. In the examples above, correctness is 3/3=1 for (1), and 2/3=0.66 for (2).

A combined measure of succinctness and correctness could be used to determine the semantic closeness of the sentences to the ideal answer. By establishing a threshold to the semantic closeness, one can find the sentences in the documents that are answers to the user's query.

The advantage of using overlap of predicates against overlap of words is that the relations between the words also affect the measure for succinctness and correctness. We can see this in the following artificial example. Let us suppose that the ideal answer to a query is:

Madrid defeated Barcelona.
defeat(x,y), madrid(x), barcelona(y)

The following candidate sentence produces the same predicates:

Barcelona defeated Madrid.
defeat(x,y), **madrid(y)**, **barcelona(x)**

However, at most two predicates only can be chosen at the same time (in boldface), because of the restrictions of the arguments. In the ideal answer, the first argument of "defeat" is Madrid and the second argument is Barcelona. In the candidate sentence, however, the arguments are reversed (the name of the variables have no effect on this). The overlap is, therefore, 2. Succinctness and correctness are 2/3=0.66 and 2/3=0.66, respectively.

5 Conclusion

We are convinced that reading comprehension tests are too difficult for the current state of art in natural language processing. Our analysis of the *Maple Syrup* story shows how much world knowledge and inference rules are needed to actually answer the test questions correctly. Therefore, we think that a more restricted kind of task that focuses rather on tractable problems than on AI-hard problems of question-answering (QA) is better suited to take our field a step further. Answer Extraction (AE) is an alternative to QA that relies mainly on linguistic knowledge. AE aims at retrieving those exact passages of a document that directly answer a given user query. AE is less ambitious than full-fledged QA since the answers are not generated from a knowledge base but looked up in the documents. These documents come from a well-defined (technical) domain and consist of a relatively small volume of data. Our test queries are real world queries that express a concrete information need. To evaluate our AE systems, we propose besides precision and recall two additional measures: succinctness and correctness. They measure the quality of answer sentences on the sentence level and are computed on the basis of the overlap of logical predicates.

To round out the picture, we address the questions in (WRC, 2000) in the view of what we said in this paper:

Q: *Can such exams [reading comprehension tests] be used to evaluate computer-based language understanding effectively and efficiently?*

A: We think that no language **understanding** system will currently be able to answer a significant proportion of such questions, which will make evaluation results difficult at best, meaningless at worst.

Q: *Would they provide an impetus and test bed for interesting and useful research?*

A: We think that the impetus they might provide would drive development in the wrong direction, viz. towards the creation of (possibly impressive) engineering feats without much linguistically interesting content.

Q: *Are they too hard for current technology?*

A: Definitely, and by a long shot.

Q: *Or are they too easy, such that simple hacks can score high, although there is clearly no understanding involved?*

A: "Simple hacks" would almost certainly score higher than linguistically interesting methods but not because the task is too simple but because it is far too difficult.

References

ADRES, 1996. *A319/A320/A321 Aircraft Maintenance Manual.* Airbus Industrie, Blagnac Cedex, France. Rev. May 1.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI International FASTUS system MUC-6 test results and analysis. In *Proc. Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland.

Otthein Herzog and Claus-Rainer Rollinger, editors. 1991. *Text Understanding in LILOG: Integrating Computational Linguistics and Artificial Intelligence - final report on the IBM Germany LILOG project*, volume 546 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Read: A reading comprehension system. In *Proc. ACL'99.* University of Maryland.

Jerry Hobbs, Douglas E. Appelt, John S. Bear, Mabry Tyson, and David Magerman. 1991. The TACITUS system: The MUC-3 experience. Technical report, AI Center, SRI International, Menlo Park, CA.

Jerry R. Hobbs, Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing.* MIT Press, Cambridge, MA.

Diego Mollá and Michael Hess. 1999. On the scalability of the answer extraction system "ExtrAns". In *Proc. Applications of Natural Language to Information Systems (NLDB'99)*, pages 219–224, Klagenfurt, Austria.

Diego Mollá and Michael Hess. 2000. Dealing with ambiguities in an answer extraction system. In *Representation and Treatment of Syntactic Ambiguity in Natural Language Processing*, Paris. ATALA.

Diego Mollá, Jawad Berri, and Michael Hess. 1998. A real world implementation of answer extraction. In *Proc. of the 9th International Conference and Workshop on Database and Expert Systems. Workshop "Natural Language and Information Systems" (NLIS'98)*, pages 143–148, Vienna, August.

MUC-7. 1998. Proc. of the seventh message understanding conference (MUC-7). http://www.muc.saic.com.

Rolf Schwitter, Diego Mollá, and Michael Hess. 1999. Extrans — answer extraction from technical documents by minimal logical forms and selective highlighting. In *Proc. Third International Tbilisi Symposium on Language, Logic and Computation*, Batumi, Georgia. http://www.ifi.unizh.ch/cl/.

Ellen M. Voorhees and Donna Harman. 1998. Overview of the seventh Text REtrieval Conference (TREC-7). In Ellen M. Voorhees and Donna Harman, editors, *The Seventh Text REtrieval Conference (TREC-7)*, number 500-242 in NIST Special Publication, pages 1–24. NIST-DARPA, Government Printing Office.

WRC. 2000. Workshop on reading comprehension texts as evaluation for computer-based language understanding systems. http://www.gte.com/AboutGTE/gto/anlp-naacl2000/comprehension.html.