RERANKING GOOGLE WITH GReG

Rodolfo Delmonte, °Marco Aldo Piccolino Boniforti

° University of Cambridge
marcoaldo.piccolinoboniforti@poste.it

Department of Language Sciences
Università Ca' Foscari – Ca' Bembo
30123, Venezia, Italy
delmont@unive.it

Abstract

We present an experiment evaluating the contribution of a system called GReG for reranking the snippets returned by Google's search engine in the 10 best links presented to the user and captured by the use of Google's API. The evaluation aims at establishing whether or not the introduction of deep linguistic information may improve the accuracy of Google or rather it is the opposite case as maintained by the majority of people working in Information Retrieval and using a Bag Of Words approach. We used 900 questions and answers taken from TREC 8 and 9 competitions and execute three different types of evaluation: one without any linguistic aid; a second one with tagging and syntactic constituency contribution; another run with what we call Partial Logical Form. Even though GReG is still work in progress, it is possible to draw clearcut conclusions: adding linguistic information to the evaluation process of the best snippet that can answer a question improves enormously the performance. In another experiment we used the actual associated to the Q/A pairs distributed by one of TREC's participant and got even higher accuracy.

1. Introduction

We present an experiment run using Google API and a fully scaled version of GETARUNS, a system for text understanding [1;2], together with a modified algorithm for semantic evaluation presented in RTE3 under the acronym of VENSES [3]. The aim of the experiment and of the new system that we called GReG (GETARUNS ReRANKS Google), is that of producing a reranking of the 10 best candidates presented by Google in the first page of a web search. Reranking is produced solely on the basis of the snippets associated to each link – two per link.
GReG uses a very "shallow" linguistic analysis which nonetheless ends up with a fully instantiated sentence level syntactic constituency representation, where grammatical functions have been marked on a totally bottom-up analysis and the subcategorization information associated to each governing predicate – verb, noun, adjective. More on this process in the sections below.
At the end of the parsing process, GReG produces a translation into a flat minimally recursive Partial Logical Form (hence PLF) where besides governing predicates – which are translated into corresponding lemmata – we use the actual words of the input text for all linguistic relations encoded in the syntactic structure.
The idea behind the experiment was this:
- given the recurrent criticisms raised against the possibility to improve web searches by means of information derived from linguistic representations we intended to test the hypothesis to the contrary;
- to this aim we wanted to address different levels of representations – syntactic and (quasi) logical/semantic, and measure their contribution if any in comparison to a simple (key) word-based computation;
- together with linguistic representation, we also wanted to use semantic similarity evaluation techniques already introduced in RTE challenges which seem particularly adequate to measure the degree of semantic similarity and also semantic consistency or non-contradictoriness of the two linguistic descriptions to compare.

The evaluation will focus on a subset of the questions used in TREC [4] made up of 900 question/answers pairs and produces the following data:
- how many times the answer is contained in the 10 best candidates retrieved by Google;
- how many times the answer is ranked by Google in the first two links – actually we will be using only snippets (first two half links);
- as a side-effect, we also know how many times the answer is not contained in the 10 best candidates and is not ranked in the first two links;
- how many times GReG finds the answer and reranks it in the first two snippets;
- how much contribution is obtained by the use of syntactic information;
- how much contribution is obtained by means of LF, which works on top of syntactic representation;
- how much contribution is obtained by modeling the possible answer from the question, also introducing some Meta operator – se use OR and the *.

Eventually, we compute accuracy measures by means of the usual Recall/Precision formula.

2. The Parser

The architecture of the parser is shown in Fig. 1 below and will be commented in this section. It is a quite common

pipeline and all the code runs in Prolog and is made up of manually built symbolic rules.
We defined our parser "mildly bottom-up" because the structure building process cycles on a procedure that collects constituents. This is done in three stages: at first chumks are built around semantic heads – verb, noun, adjective. Then prepositions and verb particles are lumped together. In this phase, also adjectives are joined to the nominal head they modify. In a third phase, sentential structure information is added at all levels – main, relative clauses, complement clauses. In presence of conjunction different strategies are applied according to whether they are coordinating or subordinating conjunctions.
An important linguistic step is carried out during this pass: subcategorization information is used to tell complements – which will become arguments in the PLF – and adjuncts apart. Some piece of information is also offered by linear order: SUBJect NPs will usually occur before the verb and OBJect NP after. Constituent labels are then substituted by Grammatical Function labels. The recursive procedure has access to calls collecting constituents that identify preverbal Arguments and Adjuncts including the Subject if any: when the finite verb is found the parser is hampered from accessing the same preverbal portion of the algorithm and switches to the second half of it where Object NPs, Clauses and other complements and adjuncts may be parsed. Punctuation marks are also collected during the process and are used to organize the list of arguments and adjuncts into tentative clauses.
The clause builder looks for two elements in the input list: the presence of the verb-complex and punctuation marks, starting from the idea that clauses must contain a finite verb complex: dangling constituents will be adjoined to their left adjacent clause, by the clause interpreter after failure while trying to interpret each clause separately.
The clause-level interpretation procedure interprets clauses on the basis of lexical properties of the governing verb. This is often non available in snippets. So in many cases, sentence fragments are built.
If the parser does not detect any of the previous structures, control is passed to the bottom-up/top-down parser, where the recursive call simulates the subdivision of structural levels in a grammar: all sentential fronted constituents are taken at the CP level and the IP (now TP) level is where the SUBJect NP must be computed or else the SUBJect NP may be in postverbal position with Locative Inversion structures, or again it might be a subjectless coordinate clause. Then again a number of ADJuncts may be present between SUBJect and verb, such as adverbials and parentheticals. When this level is left, the parser is expecting a verb in the input string. This can be a finite verb complex with a number of internal constituents, but the first item must be definitely a verb. After the (complex) verb has been successfully built, the parser looks for complements: the search is restricted by lexical information. If a copulative verb has been taken, the constituent built will be labelled accordingly as XCOMP where X may be one of the lexical heads, P,N,A,Adv.
The clause-level parser simulates the sentence typology where we may have verbal clauses as SUBJect, Inverted postverbal NPs, fronted that-clauses, and also fully inverted OBJect NPs in preverbal position.

2.1 Parsing and Robust Techniques

The grammar is equipped with a lexicon containing a list of fully specified inflected word forms where each entry is followed by its lemma and a list of morphological features, organized in the form of attribute-value pairs. However, morphological analysis for English has also been implemented and used for Out of Vocabulary (hence OOV) words. The system uses a core fully specified lexicon, which contains approximately 10,000 most frequent entries of English. Subcategorization is derived from FrameNet, VerbNet, PropBank and NomBank. These are all consulted at runtime. Eventually the semantics from the WordNet and other sources derived from the web make up the encyclopaedia. In addition to that, there are all lexical forms provided by a fully revised version of COMLEX. In order to take into account phrasal and adverbial verbal compound forms, we also use lexical entries made available by UPenn and TAG encoding. Their grammatical verbal syntactic codes have then been adapted to our formalism and is used to generate an approximate subcategorization scheme with an approximate aspectual and semantic class associated to it. Semantic inherent features for OOV words , be they nouns, verbs, adjectives or adverbs, are provided by a fully revised version of WordNet – 270,000 lexical entries - in which we used 75 semantic classes similar to those provided by CoreLex.

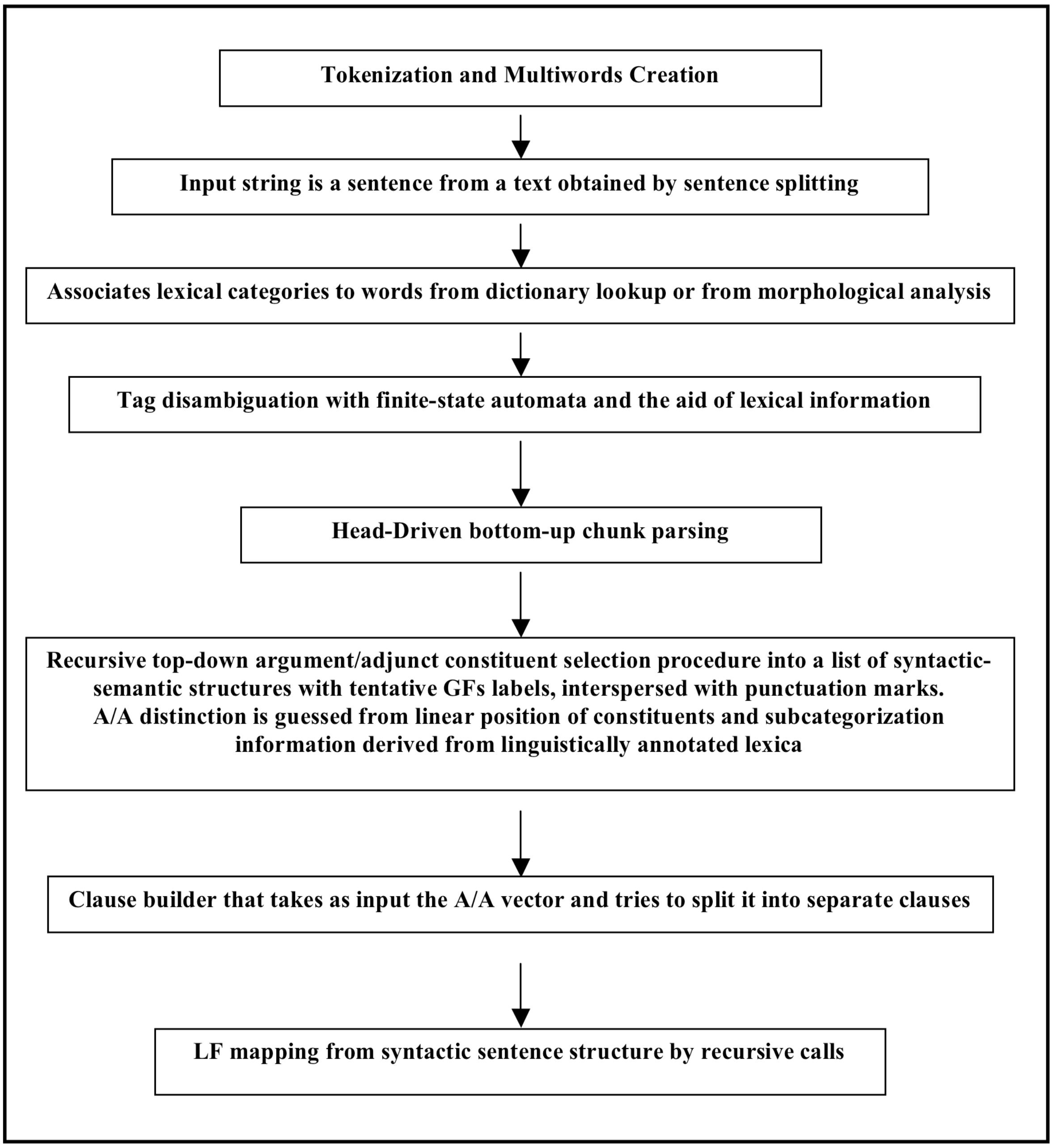
Tokenization and Multiwords Creation
Input string is a sentence from a text obtained by sentence splitting
Associates lexical categories to words from dictionary lookup or from morphological analysis
Tag disambiguation with finite-state automata and the aid of lexical information
Head-Driven bottom-up chunk parsing
Recursive top-down argument/adjunct constituent selection procedure into a list of syntactic-semantic structures with tentative GFs labels, interspersed with punctuation marks. A/A distinction is guessed from linear position of constituents and subcategorization information derived from linguistically annotated lexica
Clause builder that takes as input the A/A vector and tries to split it into separate clauses
LF mapping from syntactic sentence structure by recursive calls

Figure 1: Pipeline of parsing modules for hybrid (bottomup-topdown) version of GReG

3. The Experiment

As said above, the idea is to try to verify whether deep/shallow linguistic processing can contribute to question answering. As will be shown in the following tables, Google's search on the web has high accuracy in general: almost 90% of the answers are present in the first ten results presented to the user. However, we wanted to assume a much stricter scenario closer in a sense to TREC's tasks. To simulate a TREC task as close as possible we decided that only the first two snippets – not links - can be regarded a positive result for the user. Thus, everything that is contained in any of the following snippets will be computed as a negative result.

The decision to regard the first two snippets as distinctive for the experiment is twofold. On the one side we would like to simulate as close as possible a TREC Q/A task, where however rather than presenting precise answers, the system is required to present the sentence/snippet containing it. The other reason is practical or empirical

and is to keep the experiment user centered: user's attention should not be forced to spend energy in a tentative search for the right link. Focussing attention to only two snippets and two links will greatly facilitate the user. In this way, GReG could be regarded as an attempt at improving Google's search strategies and tools.
In order to evaluate the contribution of different levels of computation and thus get empirical evidence that a linguistically-based approach is better than a bag-of-words approach we organized the experiment into a set of concentric layers of computation and evaluation as follows:
- at the bottom level of computation we situated what we call the "default semantic matching procedure". This procedure is used by all the remaining higher level of computation and thus it is easy to separate its contribution to the overall evaluation;
- the default evaluation takes input from the first two processes, tokenization & multiword creation plus sentence splitting. Again these procedures are quite standard and straightforward to compute. So we want to assume that the results are easily reproducible as well as the experiment itself;
- the following higher level of computation may be regarded more system dependent but again it also can be easily reproduced using off-the-shelf algorithms made available for English by research centers all over the world. It regards tagging and context-free PennTreebank-like phrase-structure syntactic representation. Here we consider not only words, but word-tag pairs and word-as-head of constituent N pairs.
- the highest level is constituted by what we call Partial Logical Form, which builds a structure containing a Predicate and a set of Arguments and Adjuncts each headed by a different functor. In turn each such structure can contain Modifiers. Each PLF can contain other PLFs recursively embedded with the same structure. More on this below.
We now present three examples taken from TREC8 question/answer set, no. 3, 193, 195, corresponding respectively to ours 1,2,3. For each question we add the answer and then we show the output of tagging in PennTreebank format, then follows our enriched tagset and then the syntactic constituency structure produced by the parser. Eventually, we show the Partial Logical Form where the question word has been omitted. It can be reinserted in the analysis when the matching takes place and may appear in the other level of representation we present which is constituted by the Query in answer form passed to Google. Question words are always computed as argument or adjunct of the main predicate, so GReG will add a further match with the input snippets constituted by the conceptual substitutes of the wh-words. One substitute is visible in question no.3 when the concept "AUTHOR" is automatically added by GReG in front of the verb and after the star.

(1) What does Peugeot company manufacture? – Cars
(2) Who was the 16th President of the United States? – Lincoln
(3) Who wrote "Dubliners"? – James Joyce

Here below are the analysis where we highlight the various levels of linguistic representation relevant for our experiment only – except for the default word level:

(1) Tagging and Syntactic Constituency
what-wp, does-md, the-dt, Peugeot-nnp, company-nn, manufacture-vin, ? – pun
[what-int, does-vsup, the-art, Peugeot-n, company-n, manufacture-vin, ? - puntint]

cp-[cp-[what], f-[subj-[the, company, mod-[Peugeot]], ibar-[does, manufacture]], fint-[?]]

Partial Logical Form
pred(manufacture) arg([company, mod([Peugeot])]) adj([[], mod([[]])])

Query launched to Google API
Peugeot company manufacture *

(2) Tagging and Syntactic Constituency
who-wp, was-vbd, the-dt, 16th-cd, President-nnp, of-in, the-dt, United_States-nnp, ? – pun
[who-int, was-vc, the-art, 16th-num, President-n, of-p, the-art, United_States-n, ? - puntint]

fint-[cp-[who], ibar-[was], sn-[the, 16th, President, mod-[of, the, United_States]], fint-[?]]

Partial Logical Form
[pred(be) arg([President, mod([united, States, 16th])]) adj([])]

Query launched to Google API
United States 16th President was *

(3) Tagging and Syntactic Constituency
who-wp, wrote-vbd_vbn, "-pun, Dubliners-nns, "-pun, ? - pun
[who-int, wrote-vt, "-par, Dubliners-n, "-par, ? - puntint]

cp-[cp-[who], ibar-[wrote], fp-["], sn-[Dubliners], fp-["], fint-[?]]

Partial Logical Form
pred(write) arg([Dubliners, mod([])]) adj([])

Query launched to Google API
* author wrote Dubliners

3.1 Default Semantic Matching Procedure

This is what constitutes the closest process to the BOWs approach we can conceive of. We compare every word contained in the Question with every word contained in each snippet and we only compare content words. Stopwords are deleted.

We match both simple words and multiwords. Multiwords are created on the basis of lexical information already available for the majority of the cases. The system however is allowed to guess the presence of a multiword from the information attached to the adjacent words and again made available in our dictionaries. If the system recognizes the current word as a word starting with uppercase letter and corresponding to one of the first names listed in one of our dictionary it will try to concatenate this word to the following and try at first a match. If the match fails the concatenated word is accepted as a legitimate continuation – i.e. the name – only in case it starts by uppercase letter. Similar checking procedures have been set up for other NEs like universities, research centers, business related institutions etc. In sum, the system tries to individuate all NEs on the basis of the information stored and some heuristic inferential mechanism.

According to the type of NE we will licence a match of a simple word with a multiword in different ways: person names need to match at least the final part of the multiword, or the name institutions, locations etc. need to match as a whole.

3.2 Tags and Syntactic Heads

The second level of evaluation takes as input the information made available by the tagger and the parser. We decided to use the same approach reported in the challenges called RTE where the systems participating could present more than one run and use different techniques of evaluation. The final task was – and is – that of evaluating the semantic similarity between the question and the input snippets made available by Google. However, there is a marked difference to be taken into account and is the fact that in RTE questions where turned into a fully semantically complete assertion; on the contrary, in our case we are left with a question word to be transformed into the most likely linguistic description that can be associated with the rest of the utterance. As most systems participating in TREC challenge have done, the question has to be rephrased in order to predict the possible structure and words contained in the answer, on the basis of the question word and overall input utterance. Some of the questions contained in the TREC list do not actually constitute wh-questions (factoid or list), but are rather imperatives or iussive utterance, which tell the system – and Google – to "describe" or to "name" some linguistic item specified in the following portion of the utterance.

As others have previously done, we classify all wh-words into semantic types and provide substitute words to be place in the appropriate sentence position in order to simulate as close as possible the answer. However, this is only done in one of the modalities in which the experiment has been run. In the other modality, Google receives the actual words contained in the question.

As to experiment itself, and in particular to the matching procedure we set up, the wh- words is not used to match with the snippets. Rather we use possible linguistic items previously associated to the wh- word in a set. We also use the actual wh- words to evaluated negatively snippets containing them. In this way, we prevent similar and identical questions contained in a snippet and pointed by a link to receive a high score. We noticed that Google is unable to detect such mismatches.

We decided to use tag-word pairs in order to capture part of the contextual meaning associated to a given word. Also in the case of pairs word-as-head-of-constituent/constituent label we wanted to capture part of the contextual import of a word in a structural representation and thus its syntactic and semantic relevance in the structure. As will be clear in the following section, this is different from what is being represented in a Logical Form for how partial it may be.

3.3 Partial Logical Form and Relations

The previous match intended to compare words as part of a structure of dependencies where heads played a more relevant role than non-heads, and thus were privileged. In the higher level match what we wanted to check was the possible relations intervening between words: in this case, matching regarded two words at a time. The first and most relevant word was the PREDicate governing a given piece of PLF. The PRED can be the actual predicate governing at sentence level, with arguments and adjuncts, or it can be just the predicate of any of the Arguments/Adjuncts which in turn governed their modifiers.

Matching is at first applied to two predicates and if it succeeds, it is extended to the contents of the Argument or the Adjunct. In other words, if it is relations that this evaluation should measure, any such relations has to involve at least two linguistic elements of the PLF representation under analysis.

Another important matching procedure applied to the snippet is constituted by a check of the verbal complex. We regard the verbal compound as the carrier of semantic important information to be validated at propositional level. However, seen the subdivision of tasks, we assume that we can be satisfied by applying a partial match. This verbal complex match is meant to ascertain whether the question and the answer should be both containing a positive or a negative polarity – thus they should not convey contradictory information. It is also important to check whether the two verbal complexes are factitive or not, in that case they can contain opaque or modality operators. This second possibility needs to be matched carefully.

4. Evaluation and Conclusions

Here below we show the output of GReG in relation to one of the three questions presented above, question n.2

```
*********************************************
google7
Evaluation Score from Words and Tags : 31
Evaluation Score from Syntactic Constituent-Heads : 62
Evaluation Score from Partial Logical Form : 62
62 google8
Evaluation Score from Words and Tags : 35
Evaluation Score from Syntactic Constituent-Heads: 70
Evaluation Score from Partial Logical Form :  0
google9
Evaluation Score from Words and Tags : 33
Evaluation Score from Syntactic Constituent-Heads : 66
Evaluation Score from Partial Logical Form : 66

Snippet No.  google9
16th President of the United States ( March 4 , 1861 to April 15 , 1865 ). Nicknames : “ Honest Abe “ “ Illinois Rail - Splitter “. Born : February 12 , 1809 , . . .

Snippet No.  google7
Abraham Lincoln , 16th President of the United States of America , 1864 , Published 1901 Giclee Print by Francis Bicknell Carpenter - at AllPosters . com .

The right answer is : Lincoln

Google's best snippets containing the right answer are:
google8
Who was the 16th president of the united states ? pissaholic . . . . Abraham Lincoln was the Sixteenth President of the United States between 1861 - 1865 . . .
google7
Abraham Lincoln , 16th President of the United States of America , 1864 , Published 1901 Giclee Print by Francis Bicknell Carpenter - at AllPosters . com .

Google's best answer partially coincides with GReG.
*********************************************
```

Passing Questions to Google with GReG’s analysis produces as a result that only for 642 questions the 10 best links contain the answer. Passing Questions to Google as is produces as a result that only in 737 questions the 10 best links contain the answer. In other words, GReG’s analysis of the question that attempts at producing a model answer to use to trigger best results from Google, in fact lowers the ability of Google to search for the answer and select it in the best 10 links. This should be due to the difficulty in producing the appropriate answer form, by reordering words of the question and adding metasymbols in the appropriate positions. In fact, Google exploits also the linear order of the words contained in the question. So in case there is some mismatch the answer is not readily found or perhaps is available further down in the list of links.

	With GReG’s preanalysis	Without GReG’s anal.
Google’s 10 Best links contain the answer	755 83.01%	694 77.12%
Google’s 10 Best links do not contain the answer	145 16.9%	206 22.8%
Google Rank answer in first 2 snippets	216 28.61%	168 24.21%
Google Rank answer not in first 2 snippets	814 90.45%	803 89.23%

Table 1: Google outputs with and without the intervention of GReG’s question analysis

GReG reranks the answer in first 2 snippets	Only word match	Tagging and Syntactic heads	Partial Logical Form
With GReG’s analysis	375 58.41%	514 68.08%	543 71.92%
Without GReG’s analysis	406 55.09%	493 66.89%	495 67.16%

Table 2: GReG’s outputs at different levels of linguistic complexity

4.2 Comments

The conclusions we may safely draw is the clear improvements in performance of the system when some linguistic information is introduced in the evaluation process. In particular, when comparing the contribution of PLF to the reranking process we see that there is a clear improvement: in the case of reranking without GReG’s question analysis there is a slight but clear improvement in the final accuracy. Also, when GReG is used to preanalyse the question to pass to Google the contribution of PLF is always apparent. The overall data speak in favour of both preanalysing the question and using more linguistic processing.
If we consider Google’s behaviour to the two inputs, the one with actual questions and the one with prospective answers we see that the best results are again obtained when the preanalysis is used (28.6 vs. 24.2); also the number of candidates containing the answer increases remarkably when using GReG preprocessing (83 vs. 77).

4.3 GReG and Question-Answering from Text

In order to verify the ability of our system to extract answers from real text we organized an experiment which used the same 900 question run this time again the texts made available by TREC participants. These texts have two indices at the beginning of each line indicating respectively the question number which they should be able to answer, and the second an abbreviation containing the initial letters of the newspaper name and the date. In fact each line has been extracted by means of automatic splitting algorithms which have really messed up the whole text. In addition, the text itself has been manipulated to produce tokens which however do not in the least correspond to actual words of current orthographic forms in real newspapers. So it took us quite a lot of work to normalize the texts (5Mb.) to make them as close as possible to actual orthography.
Eventually, when we launched our system it was clear that the higher linguistic component could not possibly be used. The reason is quite simple: texts are intermingled with lists of items, names and also with tables. Since there is no principled way to tell these apart from actual texts with sentential structure, we decided to use only tagging and chunking.
We also had to change the experimental setup we used with Google snippets: in this case, since we had to manipulate quite complex structures and the choice was much more noisy, we raised our candidate set from two to four best candidates. In particular we did the following changes:

- we choose all the text stretches containing the answer/s and ranked them according to their semantic similarity;
- then we compared and evaluated these best choices with the best candidates produced by our analyses;
- we evaluated to success every time one of our four best candidates was contained in the set of best choices containing the answer;
- otherwise we evaluated to failure.

In total, we ran 882 questions because some answers did not have the corresponding texts. Results obtained after a first and only run – which took 4 days to complete on an HP workstation with 5GB of RAM, 4 Dual Core Intel processors, under Linux Ubuntu – were quite high in comparison with the previous ones, and are reported here below:

GReG finds the answer in first 4 text stretches	Tagging and Syntactic heads
Without GReG's analysis	684 / 882 77.55%

Table 3: GReG's results with TREC8/9 texts

With respect to the favourable results, we need to consider that using texts provides a comparatively higher quantity of linguistic material to evaluate and so it favours better results.

5. Conclusions

We intend to improve both the question translation into the appropriate format for Google, and the rules underlying the transduction of the Syntactic Structures into a Partial Logical Form. Then we will run the experiments again. Considering the limitations imposed by Google on the total number of questions to submit to the search engine per day, we are unable to increase the number of questions to be used in a single run.
We also intend to run GReG version for text Q/A this time with question rephrasing. We would also like to attempt using PLF with all the text stretches, excluding manually all tables and lists. We are aware of the fact that this would constitute a somewhat contrived and unnatural way of coping of unrestricted text processing. At the same time we need to check whether the improvements we obtained with snippets are attested with complete texts.
Overall, we believe to have shown the validity of our approach and the usefulness of linguistically-based evaluation methods when compared with BOWs approaches. Structural and relational information constitutes a very powerful addition to simple tagging or just word level semantic similarity measures.

6. References

Delmonte R., (2007), Computational Linguistic Text Processing – Logical Form, Semantic Interpretation, Discourse Relations and Question Answering, Nova Science Publishers, New York, ISBN: 1:60021-700-1.

Delmonte R., (2005), Deep & Shallow Linguistically Based Parsing, in A.M.Di Sciullo(ed), UG and External Systems, John Benjamins, Amsterdam/Philadelphia, pp.335-374.

Delmonte R., A. Bristot, M.A.Piccolino Boniforti, S.Tonelli (2007), Entailment and Anaphora Resolution in RTE3, in Proc. ACL Workshop on Text Entailment and Paraphrasing, Prague, ACL Madison, USA, pp. 48-53.

Litkowski, K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), The Ninth Text Retrieval Conference (TREC-9). NIST Special Publication 500-249. Gaithersburg, MD., 157-166.