Semantic Normalisation : a Framework and an Experiment

Paul Bedaride
INRIA/LORIA
Université Henri Poincaré, Nancy
paul.bedaride@loria.fr

Claire Gardent
CNRS/LORIA
Nancy
claire.gardent@loria.fr

January 21, 2009

Abstract

We present a normalisation framework for linguistic representations and illustrate its use by normalising the Stanford Dependency graphs (SDs) produced by the Stanford parser into Labelled Stanford Dependency graphs (LSDs). The normalised representations are evaluated both on a testsuite of constructed examples and on free text. The resulting representations improve on standard Predicate/Argument structures produced by SRL by combining role labelling with the semantically oriented features of SDs. Furthermore, the proposed normalisation framework opens the way to stronger normalisation processes which should be useful in reducing the burden on inference.

1 Introduction

In automated text understanding, there is a tradeoff between the degree of abstraction provided by the semantic representations used and the complexity of the logical or probabilistic reasoning involved. Thus, a system that normalises syntactic passives as actives avoids having to reason about equivalences between grammatical dependencies. Similarly, normalising phrasal synonyms into their one word equivalent (e.g., *take a turn for the worse/worsen*) or converting the semantic representation of deverbal nominals into their equivalent verbal representations (*Caesar's destruction of the city/Caesar destroyed the city*) avoids having to reason with the corresponding lexical axioms. In short, the better, semantic representations abstract away from semantically irrelevant distinctions, the less reasoning needs to be performed.

In this paper, we investigate a normalisation approach and present a framework for normalising linguistic representations which we apply to converting the dependency structures output by the Stanford parser (henceforth, Stanford Dependencies or SDs) into labelled SD graphs (LSD) that is, dependency graphs where grammatical relations have been converted to roles.

The LSD graphs we produce and the normalisation framework we present, provide an interesting alternative both for the shallow Predicate/Argument structures produced by semantic role labelling (SRL) systems and for the complex logical formulae produced by deep parsers.

Thus as we shall see in Section 2, labelled SDs are richer than the standard Predicate/Argument structures produced by SRL in that (i) they indicate dependencies between all parts of a sentence,

not just the verb and its arguments[1] and (ii) they inherit the semantically oriented features of SDs namely, a detailed set of dependencies, a precise account of noun phrases and a semantically oriented treatment of role marking prepositions, of heads and of conjunctions.

Furthermore, the normalisation framework (formal system and methodology) we present, can be extended to model and implement more advanced normalisation steps (e.g., deverbal/verbal and phrasal/lexical synonym normalisation) thereby potentially supporting a stronger normalisation process than the semantic role labelling already supported by SRL systems and by deep parsers.

In sum, although the normalised SDs presented in this paper, do not exhibit a stronger normalisation than that available in the Predicate/Argument structures already produced by deep parsers and by SRL systems, we believe that they are interesting in their own right in that they combine semantic role labelling with the semantic features of SDs. Moreover, the proposed normalisation framework opens the way for a stronger normalisation process.

The paper is structured as follows. Section 2 presents the representations handled by the system namely, the SD graphs and their labelled versions, the LSDs. Section 3 presents the rewriting system used and explains how SDs are converted to LSDs. Section 4 reports on evaluation. Section 5 discusses related work and concludes with pointers for further research.

2 (Normalised) Stanford Dependency graphs

Stanford Dependency graphs. SD graphs are syntactic dependency graphs where nodes are words and edges are labelled with syntactic relations. As detailed in [dMM06, dM08], SD graphs differ from other dependency graphs in several ways. First, they involve an extensive set of 56 dependency relations. These relations are organised in a hierarchy thereby permitting underspecifying the relation between a head and its dependent (by using a very generic relation such as *dependent*). Second, in contrast to other relational schemes such as the GR [CMB99] and PARC [KCR$^+$03], NP-internal dependency relations are relatively fine-grained[2] thereby permitting a detailed description of NPs internal structure and providing better support for an accurate definition of their semantics. Third, heads are constrained to be content words i.e., noun, verbs, adjectives, adverbs but also conjunctions. In particular, contrary to the GR scheme, SD graphs take copula *be* to be a dependent rather than a head. Fourth, SD graphs are further simplified in that some nodes may be collapsed. for instance, role marking prepositions are omitted and a trace kept of that preposition in the dependency name (e.g., *prep-on*).

The practical adequacy of SD graphs and their ability to support shallow semantic reasoning is attested by a relatively high number of experiments. Thus, in 2007, 5 out of the 21 systems submitted to the RTE (Recognising Textual Entailment) challenge used the SD representations. SDs have been used in bioinformatics for extracting relations between genes and proteins [EOR07, CS07]. It has furthermore been used for opinion extraction [ZJZ06], sentence-level sentiment analysis [MP07] and information extraction [BCS$^+$07].

[1] In the CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies [SJM$^+$08], the aim is to produce dependency structures labelled with predicate/argument relations. Although such structures are similar to the LSD graphs we produce, there are differences both in the precise type of structures built and in how these are built. We discuss this point in more detail in section 5.

[2] e.g., *appos* for apposition, *nn* for noun-noun compounds, *num* for a numeric modifier and *number* for an element in a compound number.

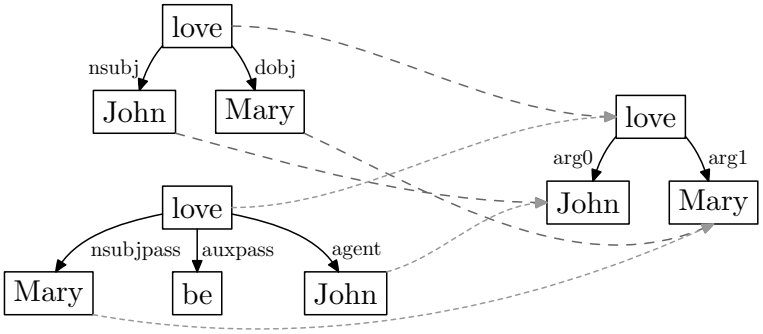

Figure 1: SDs and LSDs for "John loves Mary" and "Mary is loved by John"

Normalised Stanford Dependency graphs. From the SDs produced by the Stanford parser, we produce labelled SDs where the syntactic relations between a verb and its arguments are replaced by the roles. For instance, the SDs and LSDs for the sentences *"john loves mary"* and *"mary is loved by john"* are as given in Figure 1. The roles used in LSDs are those used in the PropBank for core- and adjunct-like arguments namely, *A0, A1, A2, A3, A4, AM* where *AM* covers all PropBank adjunct tags such as *AM-TMP, AL-LOC, etc.*.

As mentioned in the introduction, LSD graphs combine the advantages of SD graphs with semantic role labelling. From semantic role labelling, they take the more semantic predicate/argument relations. From SD graphs, they inherit the semantic oriented features such as the deletion of content poor function words, the rich hierarchy of NP internal relations and the detailed description of the relations holding between words other than the verb and its arguments.

In short, LSD graphs are both more semantic than SD graphs and richer than SRL Predicate/Argument structures.

3 Normalising dependency trees

To normalise the SD graphs, we extend the Stanford parser with a normalisation module designed to translate the grammatical relations between a verb and its arguments into roles. This normalisation module consists of an ordered set of rewrite rules and is defined semi-automatically in a two-step procedure as follows.

First, the rewrite rules for transitive verbs are defined. This first step is done manually and is based on the XTAG [Gro01] inventory of possible syntactic contexts for verbs of this type.

Second, further rewrite rules for verbs of other classes (ditransitive, verbs with sentential argument, verbs with one prepositional argument, etc.) are automatically derived from the set of rewrite rules for transitive verbs and from a small set of "base-form rewrite rules" manually defined for each class. The rules are then lexicalised using the information contained in the PropBank Frames[3].

[3]The PropBank Frames specify for each verb sense in PropBank, the arguments it accepts and the corresponding semantic roles.

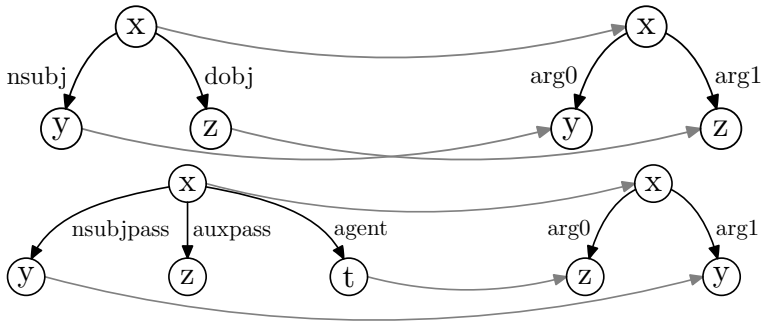

Figure 2: Rewriting rules for active and passive

3.1 Defining basic rewrite rules

In the first phase, we manually define a set of rewrite rules for each possible syntactic variation of a transitive verb.

Using the XTAG Tree Adjoining Grammar [Gro01], we start by listing these variations. Indeed a Tree Adjoining Grammar (TAG) lists the set of all possible syntactic configurations for basic clauses and groups them into so-called (tree) families. Thus the Tnx0Vnx1 family is a set of trees describing the possible syntactic contexts in which a transitive verb can occur. Further, W1nx0Vnx1 names a tree in that family which describes a syntactic context in which a transitive verb (nx0Vnx1) occurs together with a canonical nominal subject (nx0) and a questioned object (W1). We use the XTAG families to produce a list of basic clauses illustrating the possible syntactic variations of each verb type. For instance, using the Tnx0Vnx1 XTAG family, we create a "list of Tnx0Vnx1 sentences" i.e.,

(1) "John loves Mary","Mary is loved by John", "Mary, John loves", "It is Mary who is loved by John", "It is John who loves Mary", "Mary who is loved by John", "John who loves Mary",etc.

We then parse these sentences using the Stanford parser and retrieve the correct dependency structure from the output thus gathering the set of dependency structures associated by the Stanford parser with the various syntactic realisations of a given verb type.

Finally, for each distinct dependency structure found, we define a rewrite rule which maps this dependency structure onto a unique (canonical) semantic representation. For instance, the rewrite rules for the active and passive form of a sentence featuring a transitive verb are as sketched in Figure 2 (see below for the exact content of these rules).

To define our rewrite rules, we resort to a standard rewriting system namely GrGen [KG07]. Used in multiple domains (e.g., formal calculus, combinatoric algebra, operational semantics), rewriting is a technique for modelling reduction and simplification. For instance, the rewriting rule $r_1 : x * y + x * z \rightarrow x * (y + z)$ permits factorising $5 * 6 + 5 * 7 + 5 * 8$ to $5 * ((6 + 7) + 8)$. More generally, a rewriting system consists of a set of rewriting rules of the form $l \rightarrow r$ where l and r are filtering and rewriting patterns respectively. Given an object o, such a rule will apply to o if o

```
rule nx0Vnx1 {
 pattern{
  verb:element;
  if{verb.verb != "None";}
  np0:element;
  np1:element;
  verb -:nsubj-> np0;
  verb -:dobj-> np1;
 }
 replace {
  verb -:arg0-> np0;
  verb -:arg1-> np1;
 }}
```

```
rule nx1Vbynx0 {
 pattern{
  verb:element;
  if{verb.verb != "None";}
  np1:element;
  be:element;
  np0:element;
  verb -:nsubjpass-> np1;
  verb -:auxpass-> be;
  verb -:agent-> np0;
 }
 replace {
  verb -:arg0-> np0;
  verb -:arg1-> np1;
 }}
```

Figure 3: Two rewrite rules in the GrGen format

satisfies the filtering pattern l. The result of applying a rule to an object o is o where the sub-part of o matched by l is rewritten according to the rewriting pattern r. Matching consists in looking for a homograph homomorphism between the pattern graph l and the host graph h while the allowed rewriting operations include information duplication, deletion and addition[4].

In GrGen, the objects handled by rewriting are attributed typed directed multigraphs. These are directed graphs with typed nodes and edges, where between two nodes more than one edge of the same type and direction is permitted. According to its type, each node or edge has a defined set of attributes associated with it. Moreover, the type system suppports multiple inheritance on node and edge types.

Expressive and efficient, GrGen[5] is well suited to specify our normalisation rules. For instance, the rewrite rule sketched in figure 2 can be specified as given in Figure 3. The left handside (lhs) of the rule specifies a pattern in terms of nodes, node attributes, edge labels and conditions on nodes. The right handside specifies how to rewrite the subgraphs matched by the lhs.

More generally, the SD graphs can be seen as attributed typed directed multigraphs where node attributes are words and edge labels are grammatical relations. Rewrite rules can then be used to modify, add or duplicate information present in the dependency graphs to create predicate-argument structures.

Typically, rewriting is not confluent (different rule application orders yield different results)and GrGen supports various sophisticated control strategies. So far however, we simply used rule sequencing : rules are tested and fired in the order in which they are listed. They are ordered by specificity with the most specific rules listed first. For instance, the rule rewriting a long passive will precede that for a short passive thereby preventing the short passive rule from applying to a

[4]For a more precise definition of satisfaction, matching and replacement, we refer the reader to [EHK+99].

[5]There are other rewriting systems available such as in particular, the Tsurgen system used in the Stanford Parser to map parse trees into dependency graphs. We opted for GrGen instead because it fitted our requirements best. GrGen is efficient, notationally expressive (for specifying graphs but also rules and rule application strategies) and comes with a sophisticated debugging environment. Importantly, GrGen developers are also quick to react to questions and to integrate proposed modifications.

long passive sentence.

We also use GrGen "global rewriting mode". This ensures that whenever the rule filtering pattern matches several subgraphs in the input structures, the rewriting operates on each of the filtered subgraph. As we shall see in section 3, our rewrite rules are applied on not one but 5 dependency graphs at a time. Moreover the same rewrite rules may be applicable to several subgraphs in a sentence analysis (typically when the sentence contains 2 or more verbs occurring in the same syntactic configuration). Global rewriting thereby avoids having to iterate over the rule set.

3.2 Deriving new rewrite rules

Manually specifying the normalisation rules is time consuming and error prone. To extend the approach to all types of verbs and syntactic configurations, we semi-automatically derive new rewrite rules from existing ones.

Let us call *source class*, the syntactic class from which we derive new rules, *target class*, the syntactic class for which rewrite rules are being derived and *base-form rewrite rule*, a rewrite rule operating on a "base-form" that is, either on an active, a passive or a short passive form subcategorising for canonical (i.e., non extracted) arguments.

Now, let us define the set of *primitive rewrite rules* used to bootstrap the process as the set of all rewrite rules defined for the source class together with the set of base-form rewrite rules defined for the target class.

To derive new rules from the set of primitive rewrite rules, we start by computing the differences (in terms of edges, node and labels) between a source base-form rewrite rule (RR) and either a target, base-form RR ($DIFF_{+arg}$) or a source non base-form RR ($DIFF_{+movt}$). We then use the resulting *DIFFs* to compute new rewrite rules which differ either from a source RR by a $DIFF_{+arg}$ patch or from a target base-form RR by a $DIFF_{+movt}$. Figure 4 illustrates the idea on a specific example. The RR for a ditransitive verb with questioned object (*"What does John put on the table?"*, W1nx0Vnx1pnx2) is derived both by applying a $DIFF_{+W1}$ patch to the nx0Vnx1pnx2 active base-form RR (*"John put a book on the table."*) and by applying a $DIFF_{+pnx2}$ patch to the source RR operating on W1nx0Vnx1 verbs with questioned object (*"Who does Mary love?"*). Note that in this way, the same rewrite rule (W1nx0Vnx2nx1) is derived in two different ways namely, from the W1nx0Vnx1 RR by applying a $DIFF_{+pnx2}$ patch and from the nx0Vnx1pnx2 RR by applying a $DIFF_{+W1}$ one. We use this double derivation process to check the approach consistency and found that in all cases, the same rule is derived by both possible paths.

Using the method just sketched, we derived 377 rules from a set of 352 primitive rewrite rules. Although the ratio might seem weak, automating the derivation of rewrite rules facilitates system maintenance and extension. This is because whenever a correction in the set of primitive rewrite rules is carried out, the change automatically propagates to the related derived rules. In practice, we found that a real feature when adapting the system to the Propbank data. We believe that it will also be useful when extending the system to deal with nominalisations.

4 Evaluation and discussion

We evaluated our normalisation method both on a testsuite of constructed examples and on real world data namely, the Propbank corpus.

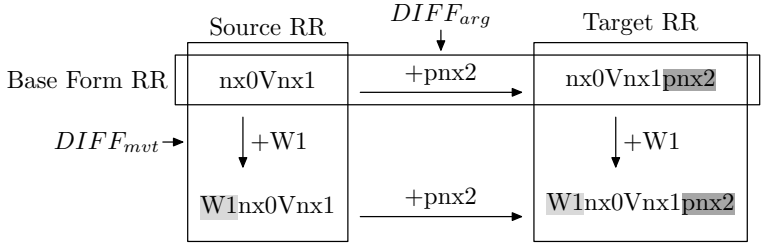

Figure 4: Deriving new rules from existing ones

4.1 Evaluation on a testsuite of constructed examples

This first evaluation aims to provide a systematic, fine grained assessment of how well the system normalises each of the several syntactic configurations assigned by XTAG to distinct verb types. The emphasis is here in covering the most exhaustive set of possible syntactic configurations possible. Because constructing the examples was intricate and time consuming, we did not cover all the possibilities described by XTAG however. Instead we concentrated on listing all the configurations specified by XTAG for 4 very distinct families namely, Tnx0Vnx1, Tnx0Vnx2nx1,Tnx0Vplnx1 and Tnx0Vnx1pnx2. The first class is the class for transitive verbs. Because of passive, this class permits many distinct variations. The second class is the class of verbs with 3 nominal arguments. This class is difficult for role labelling as the distinction between the complements often relies on semantic rather than syntactic grounds. The third class is the class of verbs with a particle and 2 nominal arguments (*ring up*) and the fourth, the class of ditransitive.

For these constructed sentences, we had no gold standard i.e., no role annotation. Hence we used logical inference to check normalisation. We proceeded by grouping the test items in (non) entailment pairs and then checked whether the associated LSDs supported the detection of the correct entailment relation (i.e., true or false).

The testsuite. Using a restricted lexicon, a set of clauses covering the possible syntactic patterns of the four verb classes and regular expressions describing sentence-semantics pairs, we develop a script generating (sentence,semantics) pairs where sentences contain one or more clauses. After having manually verified the correctness of the generated pairs, we used them to construct textual entailment testsuite items that is, pairs of sentences annotated with TRUE or FALSE dependending on whether the two sentences are related by entailment (TRUE) or not (FALSE). The resulting testsuite[6] contains 4 976 items of which 2 335 are entailments between a sentence and a clause (*1V+TE*, example 2), 1 019 between two complex sentences (*2V+TE*, example 3) and 1 622 are non-entailments (*V-TE*, example 4).

(2) T_1: John likes the book that Mary put on the table.
T_2: John likes a book
Annotations: *1V+TE*, TRUE

[6]Available at `http://www.loria.fr/~bedaride/publications/taln08-bedgar/index.html`.

(3) T_1: John likes the book that Mary put on the table.
T_2: The book which is put on the table by Mary, is liked by John
Annotations: *2V+TE*, TRUE

(4) T_1: John likes the book that Mary put on the table.
T_2: John likes a table
Annotations: *V-TE*, FALSE

Checking for entailment. For each testsuite item, we then checked for entailment by translating LSDs into FOL formulae and checking entailment between the first five LSDs derived from the parser output for the sentences contained in the testsuite item.

The translation of a LSD into a FOL formula is done as follows. Each node is associated with an existentially quantified variable and a predication over that variable where the predicate used is the word labelling the node. Each edge translates to a binary relation between the source and the target node variables. The overall formula associated with an LSD is then the conjunction of the predications introduced by each node. For instance, for the LSD given in Figure 1, the resulting formula is $\exists x, y, z : love(x) \wedge john(y) \wedge mary(z) \wedge arg0(x, y) \wedge arg1(x, z)$.

This translation procedure is of course very basic. Nonetheless, because the testsuite builds on a restricted syntax and vocabulary[7], it suffices to check how well the normalisation process succeeds in assigning syntactic variants the same semantic representation.

Results. The test procedure just described is applied to the LSD graphs produced by the normalisation module on the testsuite items. Table 5 gives the results. For each class of testsuite items (*1V+TE, 2V+TE, V-TE*), we list the percentage of cases recognised by the system as entailment (+TE) and non entailment (-TE). Because FOL is only semi-decidable, the reasoners do not always return an answer. The Failure line gives the number of cases for which the reasoners fail.

The results on positive entailments (*1V+TE*,*2V+TE*) show that the proposed normalisation method is generally successful in recognising syntax based entailments with an overall average precision of 86.3% (and a breakdown of 94.9% for *1V+TE* and 66.6% for *2V+TE* cases). Importantly, the results on negative entailments (99.2% overall precision) show that the method is not overly permissive and does not conflate semantically distinct structures. Finally, it can be seen that the results degrade for the Tnx0Vnx2nx1 class (*John gave Mary a book*). This is due mainly to genuine syntactic ambiguities which cannot be resolved without further semantic (usually ontological) knowledge. For instance, both *The book which John gave the woman* and *The woman whom John gave the book* are assigned the same dependency structures by the Stanford parser. Hence the same rewrite rule applies to both structures and necessarily assigns one of them the wrong labelling. Other sources of errors are cases where the DIFF patch used to derive a new rule fail to adequately generalise to the target structure. In such cases, the erroneous rewrite rule can be modified manually.

[7] In particular, the testsuite contains no quantifiers.

family	ans	*1V+TE*	*2V+TE*	*V-TE*
Tnx0Vnx1	+TE	585 (**98.2%**)	212 (**72.4%**)	0 (0.0%)
	-TE	11 (1.8%)	79 (27.0%)	57 (**100.0%**)
	Failure	0 (0.0%)	2 (0.6%)	0 (0.0%)
Tnx0Vnx2nx1	+TE	513 (**89.2%**)	131 (**55.7%**)	3 (0.4%)
	-TE	61 (10.6%)	103 (43.8%)	703 (**99.6%**)
	Failure	1 (0.2%)	1 (0.5%)	0 (0.0%)
Tnx0Vplnx1	+TE	567 (**95.5%**)	169 (**67.9%**)	0 (0.0%)
	-TE	27 (4.5%)	79 (31.7%)	198 (**100.0%**)
	Failure	0 (0.0%)	1 (0.4%)	0 (0.0%)
Tnx0Vnx1pnx2	+TE	550 (**96.5%**)	167 (**69.0%**)	10 (1.5%)
	-TE	16 (2.8%)	69 (28.5%)	651 (**98.5%**)
	Failure	4 (0.7%)	6 (2.5%)	0 (0.0%)
all	+TE	2215 (**94.9%**)	679 (**66.6%**)	13 (0.8%)
	-TE	115 (4.9%)	330 (32.4%)	1609 (**99.2%**)
	Failure	5 (0.2%)	10 (1.0%)	0 (0.0%)

Figure 5: Precision on constructed examples. Each cell gives the proportion of cases recognised as entailment by the system. Bold face figures give the precision i.e., the proportion of answers given by the system that are correct.

4.2 Evaluation on the PropBank

The PropBank (Proposition Bank) was created by semantic annotation of the Wall Street Journal section of Treebank-2. Each verb occurring in the Treebank has been treated as a semantic predicate and the surrounding text has been annotated for arguments and adjuncts of the predicate as illustrated in (5).

(5) [A0 He] [AM-MOD would] [AM-NEG n't] [V accept] [A1 anything of value] from [A2 those he was writing about] .

The labels used for the core and adjunct-like arguments are the following[8]. The labels A0 .. A5 designate arguments associated with a verb predicate as defined in the PropBank Frames scheme. A0 is the agent, A1 the patient or the theme. For A2 to A5 no consistent generalisation can be made and the annotation reflects the decisions made when defining the PropBank Frames scheme. Further, the AM-T label describes adjunct like arguments of various sorts, where T is the type of the adjunct. Types include locative, temporal, manner, etc.

We used the PropBank to evaluate our normalisation procedure on free text. As in the CoNLL (Conference on Natural Language Learning) shared task for SRL, the evaluation metrics used are precision, recall and F measure. An argument is said to be correctly recognised if the words spanning the argument as well as its semantic role match the PropBank annotation. Precision is the proportion of arguments predicted by a system which are correct. Recall is the proportion of correct arguments which are predicted by a system. F-measure is the harmonic mean of precision and recall. The results are given below.

[8]This is in fact simplified. The PropBank corpus additionally provide information about R-* arguments (a reference such as a trace to some other argument of A* type) and C-* arguments (a continuation phrase in a split argument).

args	0	1	2	3	4	5	a	m	total
recall	68.4%	68.2%	62.4%	47.2%	57.6%	5.3%	0.0%	64.4%	66.1%
precision	88.0%	80.2%	76.4%	83.1%	83.3%	50.0%	—	75.0%	80.6%
f-mesure	77.0%	73.7%	68.7%	60.2%	68.1%	9.5%	—	69.3%	72.6%

Precision (80.6%) is comparable to the results obtained in the ConLL 2005 SRL shared task where the top 8 systems have an average precision ranging from 76.55% to 82.28%. Recall is generally a little low (the ConLL05 recall ranged from 64.99% to 76.78%) for mainly two reasons: either the Stanford parser, did not deliver the correct analysis or the required rewrite rule was not present.

5 Conclusion

Our approach is akin to so-called semantic role labelling (SRL) approaches [CM05] and to several rewriting approaches developed to modify parsing output in RTE systems [Ass07]. It differs from the SRL approaches in that unlike most SRLs systems, it is based on a hybrid, statistic and symbolic, framework. As a result, improving or extending the system can be done independently of the availability of an appropriately annotated corpus. However, the quality, performance and coverage of the system remains dependent on those of the Stanford parser[9],

Our approach also differs from approaches that use the lambda calculus to normalise syntactic variation. In such approaches, a compositional semantics module associates words and grammar rules or derivation structures with lambda terms which in effect normalise variations such as for instance, the active/passive variation. One important advantage of lambda based approaches is that the rewriting system is confluent. The drawback however is that the specification of the appropriate lambda terms requires expert linguistic skills. In contrast, the rewrite rule approach is comparatively easier to handle (the rules presented here were developed by a computer scientist) and its use is supported by sophisticated developing environments such as GrGen which provides strong notational expressivity (the rewrite rules can include conditions, can operate on graphs of arbitrary depth, etc.), a good debugging environment and good processing times. In short although, the lambda calculus approach is undoubtly more principled, the rewrite rule approach is arguably easier to handle and easier to understand.

Normalisation of linguistic representations is not new. It is used in particular, in [BCC$^+$07, DBBT07, RTF07] for dealing with entailment detection in the RTE (Recognising Textual Entailment) challenge. The approach we present here differs from these approaches both by its systematic treatment of syntactic variation and by its use of GrGen as a framework for specifying transformations. More generally, our approach emphasises the following three points namely (i) the systematic testing of all possible syntactic variations (based on the information contained in XTAG); (ii) the use of an expressive, efficient and well-understood graph rewriting system for defining transformations; and (iii) the development of a methodology for automatically deriving new rewrite rules from existing ones.

By providing a well-defined framework for specifying, deriving and evaluating rewrite rules, we strive to develop a system that normalises NLP representations in a way that best supports

[9] [KM03] report a label F-mesure of 86.3% on section 23 of the Penn Treebank.

semantic processing. The emphasis is on aligning Predicate/Argument structures that diverge in the surface text but that are semantically similar (e.g., *John buy a car from Peter/Peter sells a car to John*). In particular, we plan to extend the system to normalise nominal dependencies (using NomBank) and converse constructions.

References

[Ass07] Association for Computational Linguistics. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic, June 2007.

[BCC+07] D. G. Bobrow, C. Condoravdi, R. S. Crouch, V. de Paiva, L. Karttunen, T. H. King, R. Nairn, L. Price, and A. Zaenen. Precision-focused textual inference. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 16–21, Prague, Czech Republic, June 2007.

[BCS+07] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI '07: Proceedings of International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India, January 2007.

[CM05] X. Carreras and L. Marquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the CoNLL-2005 Shared Task: Semantic Role Labeling*, pages 152–164, Ann Arbor, Michigan, June 2005.

[CMB99] J. Carroll, G. Minnen, and T. Briscoe. Corpus annotation for parser evaluation. In *EACL Workshop on Linguistically Interpreted Corpora*, Bergen, Norway, June 1999.

[CS07] A. B. Clegg and A. J. Shepherd. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8:24, January 2007.

[DBBT07] R. Delmonte, A. Bristot, M. A. P. Boniforti, and S. Tonelli. Entailment and anaphora resolution in rte3. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 48–53, Prague, Czech Republic, June 2007.

[dM08] M.-C. de Marneffe and C. D. Manning. The stanford typed dependencies representations. In *COLING'08 Workshop on Cross-framework and Cross-domain Parser Evaluation*, Manchester, England, August 2008.

[dMM06] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC '06: Proceedings of 5th International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy, May 2006.

[EHK+99] H. Ehrig, R. Heckel, M. Korff, Loewe M., L. Ribeiro, A. Wagner, and A. Corradini. *Handbook of Graph Grammars and Computing by Graph Transformation.*, volume 1, chapter Algebraic Approaches to Graph Transformation - Part II: Single Pushout A. and Comparison with Double Pushout A, pages 247–312. World Scientific, 1999.

[EOR07] G. Erkan, A. Ozgur, and D. R. Radev. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *EMNLP-CoNLL '07: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 228–237, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Gro01] XTAG Research Group. A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania, 2001.

[KCR+03] T. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. The parc 700 dependency bank. In *EACL workshop on Linguistically Interpreted Corpora*, Budapest, Hungary, April 2003.

[KG07] M. Kroll and R. Geiß. Developing graph transformations with grgen.net. Technical report, October 2007. preliminary version, submitted to AGTIVE 2007.

[KM03] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[MP07] A. Meena and T. V. Prabhakar. Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis. In *Ecir '07: Proceedings of 29th European Conference on Information Retrieval*, pages 573–580, Rome, Italy, April 2007.

[RTF07] A. B. N. Reiter, S. Thater, and A. Frank. A semantic approach to textual entailment: System evaluation and task analysis. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 10–15, Prague, Czech Republic, June 2007.

[SJM+08] M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL '08: Proceedings of the 12th Conference on Computational Natural Language Learning*, pages 159–177, Manchester, UK, August 2008.

[ZJZ06] L. Zhuang, F. Jing, and X.-Y. Zhu. Movie review mining and summarization. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50, Arlington, Virginia, USA, November 2006. ACM.