Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories

Groningen, The Netherlands
January 23-24, 2009

Editors:
Frank Van Eynde
Anette Frank
Koenraad De Smedt
Gertjan van Noord

Published by
LOT
Janskerkhof 13
3512 BL Utrecht
The Netherlands

phone: +31 30 253 6006
fax: +31 30 253 6406
e-mail: lot@let.uu.nl
http://www.lotschool.nl

Cover illustration: ships at Hoge Der Aa, Groningen. Photo by Çağrı Çöltekin.

ISBN 978-90-78328-77-3
NUR 616

Preface

The Seventh International Workshop on Treebanks and Linguistic Theories (TLT7) is held in Groningen (the Netherlands) on January 23-24, 2009, see `http://www.let.rug.nl/tlt`. Like its predecessors, it provides a forum for researchers in the field of Computational Linguistics who are experts in the design, creation and exploitation of treebanks and their relation to linguistic theories.

Treebanks have become crucial for the development of data-driven approaches to natural language processing, human language technologies, grammar extraction, and linguistic research in general. Manifold projects aim at compiling representative treebanks for specific languages. Other projects focus on the development of tools for exploration of annotated treebanks, or explore annotation beyond syntactic structure and beyond single languages.

The call for papers for TLT7 asked for unpublished, completed work. 27 submissions were received, 23 for full papers, 4 for poster/demo presentations. The submissions were authored by researchers from 15 different countries in America, Asia and Europe. Every submission was reviewed by at least three reviewers. Reviewers were from eight different countries. Based on their scores and the comments they provided on the content and quality of the papers, 13 papers and 3 posters/demos were accepted for presentation and publication, which corresponds to an acceptance rate of 59,3 %. Together the accepted submissions cover a wide range of topics, including the building, querying, exploring, exploiting and evaluating of treebanks.

Completing the program are the invited lectures by Robert Malouf (San Diego State University) and Adam Przepiórkowski (Polish Academy of Sciences).

Words of gratitude and acknowledgement are first of all due to the local organizers and hosts, and the STEVIN Lassy project which took the initiative for the organization. They have done a great job during the preparation of this workshop and will certainly make our stay in Groningen worthwhile and pleasant.

We also gratefully mention the financial support from Textkernel, TST-centrale, Stevin, CLCG, Gridline and Q-Go.

Finally, we thank the Editorial Board of the Netherlands Graduate School of

Linguistics (LOT) for accepting to publish the TLT7 Proceedings in the LOT Occasional Series.

The TLT7 Co-Chairs

Frank Van Eynde
University of Leuven

Anette Frank
University of Heidelberg

Koenraad De Smedt
University of Bergen

Program Committee

Chairs:
Frank Van Eynde, University of Leuven, Belgium
Anette Frank, University of Heidelberg, Germany
Koenraad De Smedt, University of Bergen, Norway

Members:
Anne Abeillé, France
Gosse Bouma, the Netherlands
Aoife Cahill, Germany
Stefanie Dipper, Germany
Josef van Genabith, Ireland
Jan Hajič, Czech Republic
Erhard Hinrichs, Germany
Julia Hockenmaier, USA
Sandra Kübler, USA
Domen Marinčič, Slovenia
Yuji Matsumoto, Japan
Detmar Meurers, USA
Yusuke Miyao, Japan
Joakim Nivre, Sweden
Stephan Oepen, Norway
Adam Przepiórkowski, Poland
Victoria Rosén, Norway
Yvonne Samuelsson, Sweden
Kiril Simov, Bulgaria
Manfred Stede, Germany
Yannick Versley, Germany

Organizing Committee

Chair:
Gertjan van Noord, University of Groningen, the Netherlands

Members:
Gosse Bouma
Barbara Plank
Tim van de Cruys
Jelena Prokić
Çağrı Çöltekin
Erik Tjong Kim Sang, all University of Groningen, the Netherlands
Ineke Schuurman, University of Leuven, Belgium

Proceedings of previous TLT workshops and invited speakers

E. Hinrichs & K. Simov (eds.), Proceedings of the First Workshop on Treebanks and Linguistic Theories, Sozopol (Bulgaria), September 20-21, 2002. v + 274 pages. `http://www.bultreebank.org/Proceedings.html`

J. Nivre & E. Hinrichs (eds.), Proceedings of the Second Workshop on Treebanks and Linguistic Theories, Växjö (Sweden), November 14-15. Växjö University Press, 2003. 232 pages.
Invited speakers: Thorsten Brants (Google Inc.), Stephan Oepen (U. of Oslo).

S. Kübler, J. Nivre, E. Hinrichs & H. Wunsch (eds.), Proceedings of the Third Workshop on Treebanks and Linguistic Theories, Tübingen (Germany), December 10-11, 2004. Eberhard Karls Universität Tübingen, Seminar für Sprachwissenschaft. vi + 203 pages.
Invited speakers: Fred Karlsson (U. of Helsinki), Collin Baker (ICSI, Berkeley).
Publication of selected papers in: E. Hinrichs & K. Simov (eds.): Treebanks and Linguistic Theories. Special Issue of the Journal on Research on Language and Computation. Vol. 2, Nr. 4, 2004.

M. Civit, S. Kübler & M.A. Martí (eds.), Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories, Barcelona (Spain), December 9-10, 2005. Universitat de Barcelona, Publicacions i Edicions. 220 pages.
Invited speakers: Frank van Eynde (U. Leuven), Manfred Pinkal (U. of the Saarland).

J. Hajič & J. Nivre (eds.), Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories, Prague (Czech Republic), December 1-2, 2006. Univerzita Karlova v Praze, Ústav Formální a Aplikované Lingvistiky. 258 pages.
Invited speakers: Gosse Bouma (U. of Groningen), Martha Palmer (U. of Colorado at Boulder).

K. De Smedt, J. Hajič & S. Kübler (eds.), Proceedings of the Sixth International

Workshop on Treebanks and Linguistic Theories, Bergen (Norway), December 7-8, 2007. Northern European Association for Language Technology Proceedings Series, Vol. 1.viii + 218 pages. http://dspace.utlib.ee/dspace/handle/10062/4476.
Invited speakers: Erhard Hinrichs (U. of Tübingen), Julia Hockenmaier (U. of Illinois).

Contents

Linguistic Annotation for Valence Acquisition and for its Evaluation

Adam Przepiórkowski
Polish Academy of Sciences
Institute of Computer Science
E-mail: adamp@ipipan.waw.pl

Abstract

Valence acquisition is the task consisting in the automatic extraction (learning) of subcategorisation – or argument structure – from corpora. In this talk I will concentrate on two issues. The first issue is: how much linguistic annotation do we need for valence acquisition? Approaches range from linguistically lean, e.g., Brent's 1993 proposal to infer valence information from co-ocurrences of verbs with pronouns and functional words, to more recent proposals to read valence off from lavishly annotated treebanks. I will present the results of some experiments from Polish suggesting that shallow (or partial) parsing may be as useful in this task as more difficult and less efficient deep parsing. The second issue concerns the evaluation of the results of automatic valence acquisition. The common methodology is to compare the automatic results to a manually constructed valence dictionary, but – again on the basis of some experiments carried out for Polish – I will point out various weaknesses of this methodology and argue for the more costly and, hence, less common corpus-based evaluation.

Treebanks and Evolutionary Simulation for Explaining Typological Patterns

Robert Malouf
San Diego State University
Department of Linguistics & Asian / Middle Eastern Languages
E-mail: rmalouf@mail.sdsu.edu

Abstract

Recent work in Evolutionary Phonology (Blevins 2005, 2006, Blevins & Wedel 2008, Yu 2007, among others) has developed alternate explanations for typological universals or tendencies found across the sound systems of unrelated languages. This research emphasizes the role of patterns of language use and language change in the development of cross-linguistic patterns, rather than placing the burden of explanation on synchronic cognitive factors (i.e., Universal Grammar).

In this talk, we will review extensions of this work into the domain of morphology (Ackerman, Blevins, and Malouf to appear). We investigate the Paradigm Cell Filling Problem, a particular question for inflectional morphological systems which has received relatively little attention in the theoretical literature. Specifically, we ask: How do speakers of morphologically complex languages predict the full inflectional (or derivational) paradigms of novel words, given exposure to a small number of surface word forms? For example, a noun in Tundra Nenets can appear in 210 different inflected forms. Given exposure to one of these forms of a novel noun, how does a Tundra Nenets speaker predict the other 209 forms?

Our hypothesis is that speakers' need to solve the Paradigm Cell Filling Problem serves as a strong evolutionary pressure on language, which in turn leads morphological systems to develop in particular directions. Thus the Paradigm Cell Filling Problem is an indirect explanation for some of the typological patterns found cross-linguistically in morphological systems. In order to test our hypotheses about language development, we perform computer simulations of language evolution across many generations, to see which factors cause which patterns to arise. In this way, we treat language as a complex adaptive system and therefore link linguistic study to larger trends in the biological and social sciences (e.g., Miller and Page 2007).

As with many other complex adaptive systems, the outcome of our simulations of linguistic evolution can be highly dependent on the initial condi-

tions. Therefore, to be reliable, our simulations need to be based on detailed and accurate information about synchronic language states. As the domain of investigation moves from morphophonology to morphosyntax, it becomes more difficult to find the information we need in conventional typological databases and lexicons. The kind of information we need can only be found in treebanks – corpora with highly detailed linguistic annotations. Therefore, the development of large, richly annotated corpora in a variety of typological diverse languages is crucial to the evolutionary program for explaining cross-linguistic evolutionary patterns.

A Testsuite for Testing Parser Performance on Complex German Grammatical Constructions

Sandra Kübler
Indiana University
Department of Linguistics
skuebler@indiana.edu

Ines Rehbein
Universität des Saarlandes
Computational Linguistics
rehbein@coli.uni-sb.de

Josef van Genabith
Dublin City University
School for Computing
josef@computing.dcu.ie

1 Introduction

Traditionally, parsers are evaluated against gold standard test data. This can cause problems if there is a mismatch between the data structures and representations used by the parser and the gold standard. A particular case in point is German, for which two treebanks (TiGer and TüBa-D/Z) are available with highly different annotation schemes for the acquisition of (e.g.) PCFG parsers. The differences between the TiGer and TüBa-D/Z annotation schemes make fair and unbiased parser evaluation difficult [7, 9, 12]. The resource (TEPACOC) presented in this paper takes a different approach to parser evaluation: instead of providing evaluation data in a single annotation scheme, TEPACOC uses comparable sentences and their annotations for 5 selected key grammatical phenomena (with 20 sentences each per phenomena) from both TiGer and TüBa-D/Z resources. This provides a 2 times 100 sentence comparable testsuite which allows us to evaluate TiGer-trained parsers against the TiGer part of TEPACOC, and TüBa-D/Z-trained parsers against the TüBa-D/Z part of TEPACOC for key phenomena, instead of comparing them against a single (and potentially biased) gold standard. To overcome the problem of inconsistency in human evaluation and to bridge the gap between the two different annotation schemes, we provide an extensive error classification, which enables us to compare parser output across the two different treebanks.

In the remaining part of the paper we present the testsuite and describe the grammatical phenomena covered in the data. We discuss the different annotation strategies used in the two treebanks to encode these phenomena and present our error classification of potential parser errors.

2 TEPACOC - The Testsuite

TEPACOC contains 200 sentences carefully selected from two German treebanks. The sentences cover five complex grammatical constructions which are extremely difficult for a PCFG parser to process:

1. PP Attachment: Noun (PPN) vs. Verb Attachment (PPV)
2. Extraposed Relative Clauses (ERC)
3. Forward Conjunction Reduction (FCR)
4. Subject Gap with Finite/Fronted Verbs (SGF)
5. Coordination of Unlike Constituents (CUC).

PP attachment is the canonical case of structural ambiguity and constitutes one of the major problems in (unlexicalised) parsing, since disambiguation often requires lexical rather than structural information [5]. The testsuite allows us to investigate which of the different encoding strategies in the two treebanks is more successful in resolving PP attachment ambiguities.

The second construction we included in TEPACOC are extraposed relative clauses. According to Gamon et al. [2], who present a case study in German sentence realisation, 35% of all relative clauses in a corpus of German technical manuals are extraposed, while in a comparable corpus of English technical manuals less than one percent of the relative clauses have been subject to extraposition. This shows that extraposed relative clauses are a frequent phenomenon in German and worthwhile to be considered for parser evaluation.

Coordination is a phenomenon which poses a great challenge not only to statistical parsing but also to linguistic theories in general (see for example [6, 13, 11, 15] for a discussion on different types of coordination in LFG, HPSG, GPSG and CCG). Harbusch and Kempen [4] present a corpus study on the TiGer treebank (Release 2), where they investigate cases of clausal coordination with elision. They found 7196 sentences including clausal coordinations, out of which 4046 were subject to elisions. 2545 out of these 4046 sentences proved to be Forward Conjunction Reduction, and 384 sentences contained Subject Gaps with Finite/Fronted Verbs. We included FRC and SGF as the most frequent forms of non-constituent coordination in the testsuite. The TiGer treebank (Release 2) contains 381 sentences with at least one CUC, which means that coordination of unlike constituents are as frequent as SGF. Additionally, we choose CUC to be part of the TEPACOC because, from a linguistic point of view, they are quite interesting and put most linguistic theories to the test. The testsuite is available as a list of sentence numbers referring to the original treebanks so that interested parties can extract the sentences.[1]

[1] `http://jones.ling.indiana.edu/~skuebler/tepacoc`

2.1 Data Sources: TiGer and TüBa-D/Z

The data for the corpus comes from two different sources: the TIGER treebank (Release 2) [1] and the TüBa-D/Z (Release 3) [16]. Both treebanks contain German newspaper text and are annotated with phrase structure and dependency (functional) information. While both treebanks employ the same POS Tag Set (STTS) [14], the number of category labels and grammatical function labels varies. The most important differences between the two treebanks are: (1) the annotation in TIGER is rather flat compared to the more hierarchical annotation in TüBa-D/Z, (2) TIGER does not annotate unary branching, (3) TüBa-D/Z annotates topological fields, and (4) long distance dependencies in TiGer are expressed via crossing branches while in TüBa-D/Z, the same phenomenon is expressed with the help of grammatical function labels.

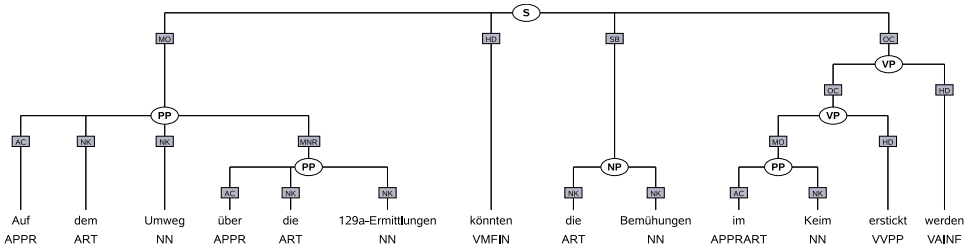
For each of the 5 grammatical phenomena listed above, we selected 20 sentences with a lenght ≤ 40 from each TIGER and TüBa-D/Z. This results in a test set of 200 sentences, 100 from each treebank. Below we give a survey of the testsuite: we describe the annotation of the phenomena in TEPACOC and discuss the different annotation decisions made in TIGER and TüBa-D/Z. The differences in treebank design do not support a systematic description of different error types like e.g. span errors, attachment errors or grammatical function label errors, as the same phenomenon might be encoded with the help of GF labels in one treebank and by using attachment in the other treebank. Therefore, we present a descriptive error classification scheme based on empirical data, capturing all potential parser errors on the specific grammatical phenomena.

2.2 PP Attachment: Noun (PPN) vs. Verb Attachment (PPV)

PP attachment is one of the problems discussed most in parsing since a correct attachment often requires lexical rather than purely structural information. In TIGER, noun attachment results in a flat tree structure in which the PP is attached on the same level as the head noun, while verb attachment has the PP grouped under the VP or the S node. Both NP and PP attachment are present in the TiGer example (1).[2]

In TüBa-D/Z, NP postmodifiers are attached on a higher level once the NP is grouped. For verb attachment the PP is directly attached to the governing topological field, the functional label shows whether it is considered a prepositional object (OPP), an optional prepositional object (FOPP), an unambiguous verbal modifier (V-MOD), or an ambiguous one (MOD). (2) shows a TüBa-D/Z representation of NP and VP attachment.

[2] Some of the examples have been shortened for readability.



(1) Auf dem Umweg **über die 129a-Ermittlungen** könnten die Bemühungen **im Keim** erstickt
By the detour via the 129a-investigations could the efforts in the bud nipped
werden.
be.
"With the 129a investigations, the efforts of the autonomouns activists could be nipped in the bud."

Error description		TIGER / TüBa-D/Z
(A)	correct GF & correct head of PP, span incorrect	
(B)	correct span, incorrect GF	
(C)	incorrect span, incorrect GF	
(D)	wrong attachment	

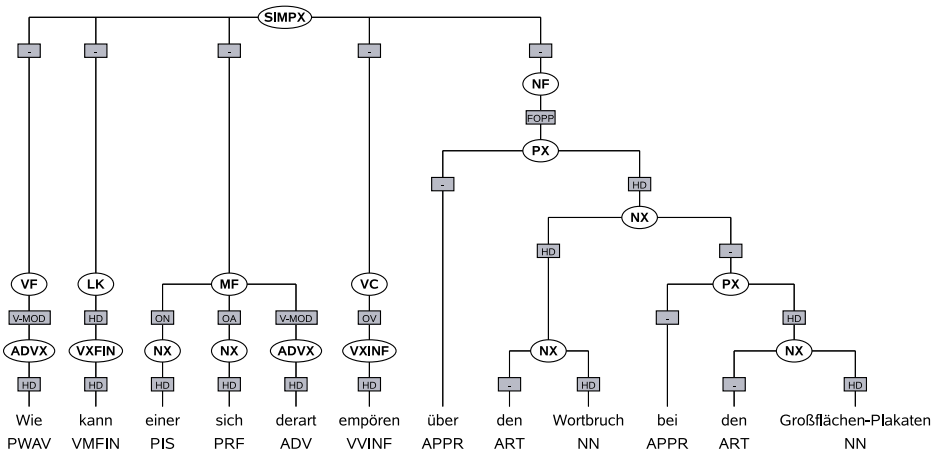
Table 1: Error classification for PP attachment

2.2.1 Error Classification (PPN vs. PPV)

We consider PP attachment parsed correctly if the PP is recognized correctly and if it is attached correctly with the correct grammatical function (Table 1). In TüBa-D/Z, extraposed PPs that are extracted from a preceding NP are not attached directly to the NP, their attachment is indicated in the grammatical function label. If an extraposed PP is attached incorrectly, the GF label is incorrect. In such cases, error code D must be used.

2.3 Extraposed Relative Clauses (ERC)

Extraposed relative clauses in German are treated as adjuncts to the head noun they modify, but there is no agreement in the literature whether they are base-generated locally [3] or get their final position through movement [10]. In TIGER, relative clauses are attached to the mother node of the head noun, which results in crossing branches for extraposed clauses, as in (3). The relative clause has the categorial node label S and carries the GF label RC. The relative pronoun is attached directly to the S node.



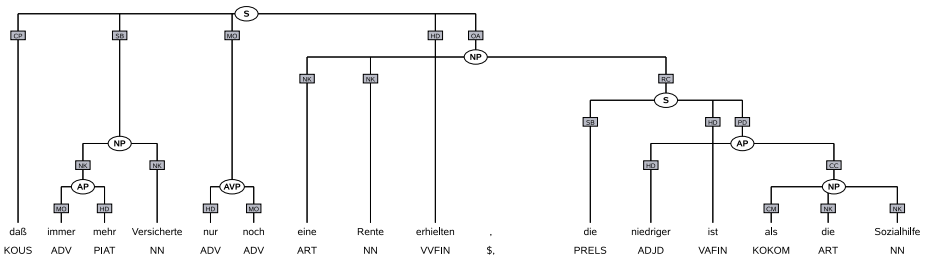
(2) Wie kann einer sich derart empören **über den Wortbruch bei den**
How can one *refl.* so revolt about the breach of promise concerning the
Großflächen-Plakaten?
large-scale posters?
"How can someone bristle at the breach of promise concerning the large-scale posters?"

In TüBa-D/Z, the extraposed relative clause is located in the final field (NF) and is associated with the node label R-SIMPX. The grammatical function label references the head noun modified by the relative clause, as in (4). The relative pronoun is embedded inside of an NP (NX) which is attached to a C node (complementizer for verb-final sentences).

2.3.1 Error Classification (ERC)

We consider an extraposed relative clause parsed correctly if the clause has been identified by the parser as a relative clause and is associated with the correct head noun, and if the phrase boundaries have been recognized correctly. Due to differences in annotation, here we have to adapt the error analysis to the annotation scheme of each treebank. Table 2 shows our error classification for extraposed relative clauses with an error specification for each treebank.

In TIGER, the grammatical function label carries the information that the clause is a relative clause while in TüBa-D/Z, the same information is encoded in the categorial node label. Therefore, error description (A) corresponds to a function label error in TIGER and to a categorial node label error in TüBa-D/Z. The relationship between the relative clause and its head noun is expressed through attachment in TIGER and by the use of a GF label in TüBa-D/Z. Therefore (B) is caused by a



(3) ... dass immer mehr Versicherte nur noch eine **Rente** erhielten, **die niedriger ist als die Sozialhilfe**
... that always more insurants just still a pension would receive, which lower is than the social welfare

"... that more and more insurants receive a pension lower than social welfare"

Error description	TIGER	TüBa-D/Z
(A) Clause not recognized as rel. cl.	Grammatical function incorrect	SIMPX label instead of R-SIMPX
(B) Head noun incorrect	Attachment error	Grammatical function incorrect
(C) Clause not recognized	Clause not recognized	Clause not recognized
(D) Clause boundaries not correct	Span error	Span error

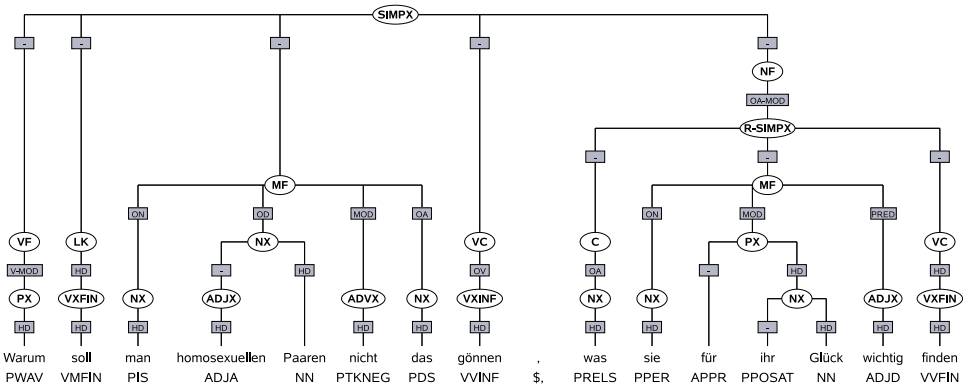
Table 2: Error classification for extraposed relative clauses

wrong attachment decision in TIGER and by a GF label error in TüBa-D/Z. For (C) the parser fails to identify the relative clause alltogether. This is usually caused by a POS tagging error, i.e. when the parser fails to assign the correct POS tag to the relative pronoun. (D) applies to both annotation schemes: here, the main components of the clause have been identified correctly but the phrase boundaries are slightly wrong.

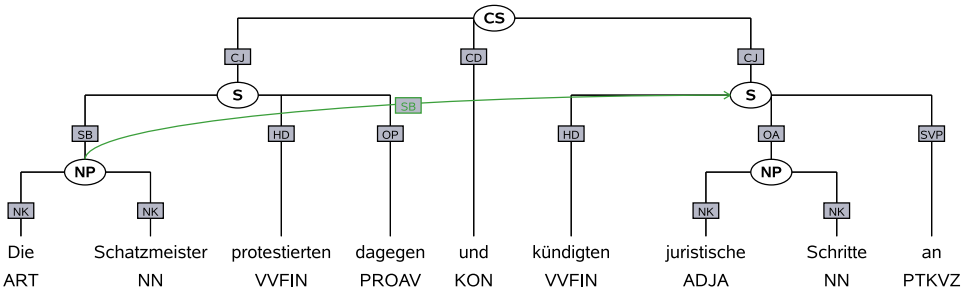
2.4 Forward Conjunction Reduction (FCR)

Forward Conjunction Reduction is a form of coordination in which both conjuncts contain a main verb and its arguments, and in which the conjuncts share the left peripheral context. In TIGER, the coordination is on sentence level, as in (5). The left peripheral context and the first conjoined verb phrase are grouped as a clause (S), and the second conjunct is projected to an elliptical clause. Both clauses are then coordinated. The role of the left peripheral context in the second clause is annotated via a secondary edge.

In TüBa-D/Z, the coordination is on the level of field combinations, as in (6).



(4) Warum soll man homosexuellen Paaren nicht **das** gönnen, **was sie für ihr Glück wichtig finden?**
Why shall one homosexual couples not that grant, which they for their luck important find?
"Why shouldn't homosexual couples be granted what they think is important to their happiness."

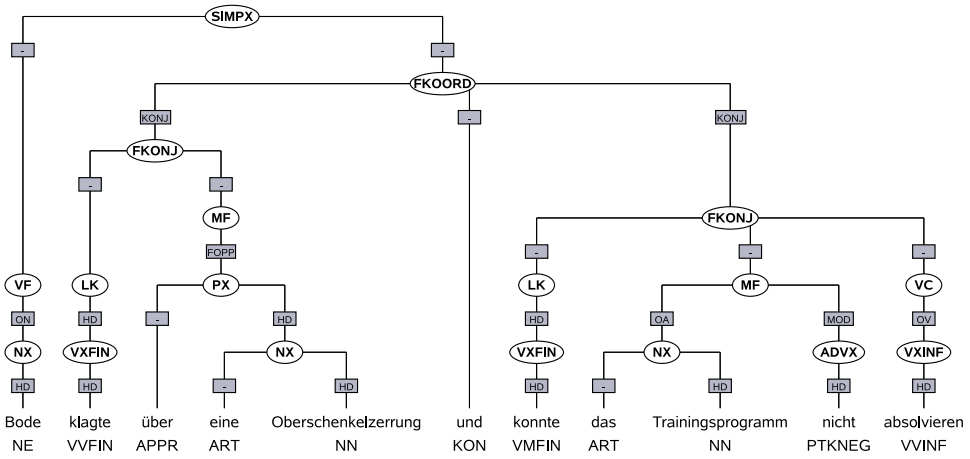


(5) Die Schatzmeister protestierten dagegen und kündigten juristische Schritte an.
The treasurers protested against it and announced legal action *verb part.*
"The treasurers protested and announced, they would take legal action."

As a consequence of the field model, the left peripheral context constitutes the initial field (VF) and is attached only once the coordination is grouped. Within the coordination, each conjunct is a combination of the verbal field (LK or VC) and its arguments (MF).

2.4.1 Error Classification (FCR)

We consider the FCR parsed correctly if the parser has identified the coordination, has assigned the subject label to the appropriate node, and if no other node in the



(6) Bode klagte über eine Oberschenkelzerrung und konnte das Trainingsprogramm nicht
Bode complained about a strain of the thigh and could the training regime not
absolvieren.
finish.
"Bode complained about a strain of the femural muscle and could not finish the training."

first or second constituent has been associated with the subject label. Here the annotation schemes allow us to use the same error specification for both treebanks (Table 3).

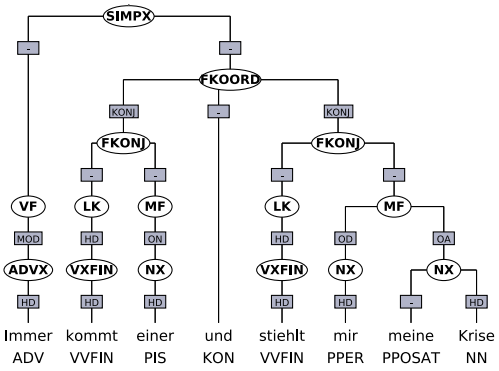
Error description		TIGER / TüBa-D/Z
(A)	Parser incorrectly annotates subject in one of the constituents	
(B)	Parser fails to identify subject	
(C)	Coordination not recognized	
(D)	Second subject in first conjunct	
(E)	Span error	(only in TüBa-D/Z)

Table 3: Error classification for forward conjunction reduction

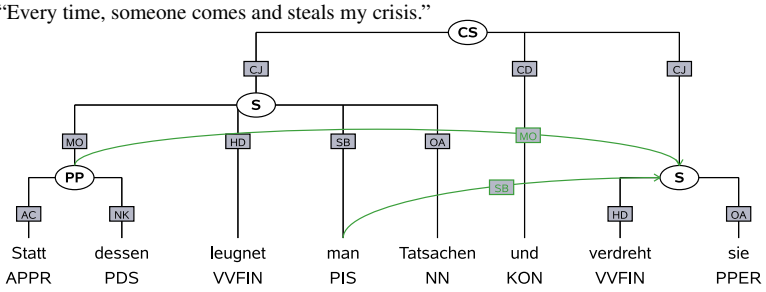
2.5 Subject Gap with Fronted/Finite Verbs (SGF)

Next, we discuss the case of asymmetric coordination where the subject of the left conjunct is realized in the middle field, while in the right conjunct the subject is missing. In TüBa-D/Z, subject gapping is treated as a complex coordination of fields (FKOORD), as in (7). The subject is realized in the middle field of the first constituent and has the functional label ON (nominative object). Both constituents are associated with the functional label FKONJ (conjunct with more than one field).

In TIGER, subject gaps with fronted/finite verbs are encoded as a coordination



(7) Immer kommt **einer** und stiehlt mir meine Krise.
Always comes someone and steals me my crisis.
"Every time, someone comes and steals my crisis."



(8) Statt dessen leugnet **man** Tatsachen und verdreht sie.
Instead denies one facts and twists them.
"Instead, the facts are denied and twisted."

of sentences (CS), as in (8). As in TüBa-D/Z, the subject is realized in the first constituent and can be identified by the grammatical function label SB (subject). With the help of labeled secondary edges (SB), TIGER encodes explicitly that the subject of the first constituent should also be interpreted as the subject of the second constituent.

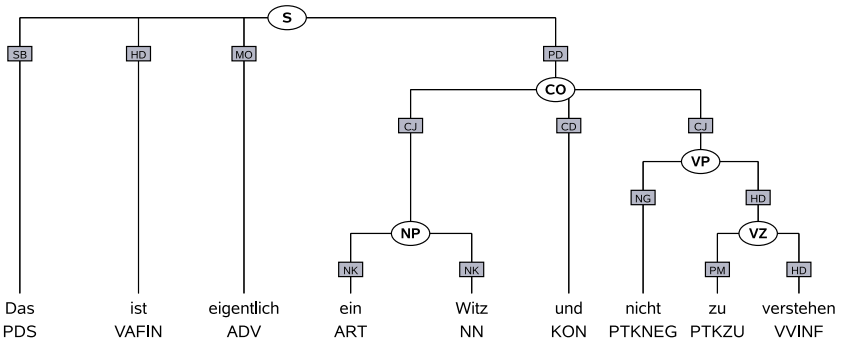
2.5.1 Error Classification (SGF)

We consider the subject gap construction parsed correctly if the parser has identified the coordination, has assigned the subject label to the right node in the first constituent, and no other node in the first or second constituent has been associated with the subject label. Here, the annotation schemes allow us to use the same error specification for both treebanks (Table 4).

Error description		TIGER / TüBa-D/Z
(A)	Parser incorrectly annotates subject in second conjunct	
(B)	Parser fails to identify subject in first conjunct	
(C)	Coordination not recognized	
(D)	Parser annotates additional subject in first conjunct	
(E)	Parser fails to identify the verb in the sentence	

Table 4: Error classification for subject gap with fronted/final verb

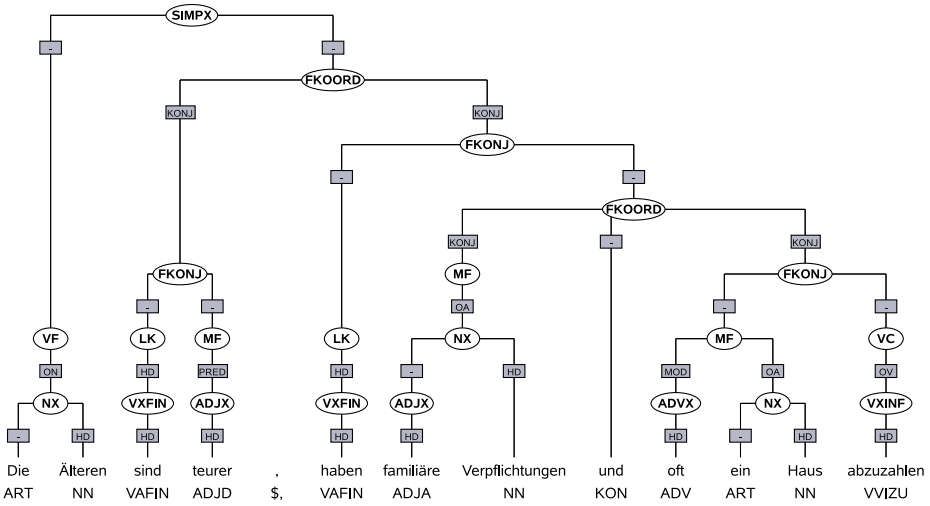
2.6 Coordination of Unlike Constituents (CUC)



(9) Das ist eigentlich ein Witz und nicht zu verstehen.
This is actually a joke and not to understand.
"This actually is a joke and hard to understand."

This section covers three types of coordinations of unlike constituents: VPs coordinated with adjectival phrases (AP), VPs coordinated with NPs, and clauses (S) coordinated with NPs. Here, we will concentrate on the second type since it shows the greatest differences between the two annotations schemes: in TIGER, the coordination is rather straightforward, the VP and the NP project to a coordinated phrase (CO), as in (9). Since the functional labels for the conjuncts (CJ) describe their conjunct status and the functional label of the coordination is the same as that associated with verb phrase (OC), the annotation does not contain explicit information which grammatical function the NP performs in the clause.

In TüBa-D/Z, the coordination is on the field level and the VP is represented as a combination of the verbal field and the middle field (MF), as in (10). The NP is projected to the MF, too, before both conjuncts are coordinated. In this case, the individual grammatical functions are retained in the constituents under the MFs.



(10) Die Älteren sind teurer, haben familiäre Verpflichtungen und oft ein Haus abzuzahlen.
The elderly are more expensive, have familial committments and often a house to repay.
"The elderly are more expensive, have family committments and often have to pay off a house."

2.6.1 Error Classification (CUC)

Since the two annotation schemes differ drastically in the annotation of coordinations of unlike constituents, we decided to use a correct/incorrect distinction only. A CUC is considered correct if the constituents are recognized with correct spans and correct heads.

2.7 Discussion

The phenomena described above are hard nuts to crack for a PCFG parser, and occur quite frequently in German text. The testsuite allows us (1) to assess the performance of constituent-based parsers on the grammatical constructions described above, and (2) to accomplish a fine-grained investigation on the impact of specific treebank design decisions on parser performance for specific syntactic phenomena. We showed that the two German treebanks choose different ways to encode the same syntactic phenomena. TEPACOC provides an additional means to answer questions like the following:

- What is more suitable to disambiguate PP attachment: the flat trees in TiGer or the more hierarchical annotation in TüBa-D/Z?
- Do topological fields support the identification of coordinations?

- Which of the two annotation strategies is more adequate to resolve non-local dependencies, as in ERC, FCR and SGF constructions?

Kübler et al. [8] put the TEPACOC to the test and compare results for constituent-based and dependency-based automatic evaluation measures with a manual evaluation on the TEPACOC sentences. They show that constituent-based evaluation measures are highly biased towards the more hierarachical annotation scheme of the TüBa-D/Z, while a dependency-based evaluation gives better results for labelled accuracy for parsers trained on the flat structures of the TiGer treebank. The dependency-based evaluation is backed up by a manual evaluation on the TEPACOC sentences, which sheds some light on the underlying reasons for the difference in parser performance on the two treebanks: (1) TiGer benefits from the flat annotation which makes it more transparent for the parser to detect constructions like ERC, FCR and SGF; (2) TüBa-D/Z suffers from the more hierarchical structure where relevant clues are embedded too deep in the tree for the parser to make use of it; (3) the additional layer of topological fields in TüBa-D/Z increases the number of possible attachment positions (and so of possible errors); and (4) topological fields reduce the number of rules in the grammar and improve the learnability especially for small training sets.

3 Conclusion and Future Work

We presented TEPACOC, a corpus for testing parser performance on complex grammatical constructions. TEPACOC covers five grammatical phenomena and provides a well-defined error categorisation which enables us to observe the influence of treebank design on specific grammatical constructions and so gain valuable insights for the future development and standardisation of language resources.

At the moment, TEPACOC includes German data only, but it can easily be extended to other languages. It is understood that the limited size of the testsuite does challenge the representativeness of the results. Therefore, TEPACOC should be used in addition to other evaluation metrics, providing an additional means to assess parser performance on a linguistic level and enabling us to compare results across different annotation schemes and languages.

References

[1] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In *Proceedings of TLT 2002*, Sozopol, Bulgaria, 2002.

[2] Michael Gamon, Eric Ringger, Zhu Zhang, Robert Moore, and Simon Corston-Oliver. Extraposition: a case study in German sentence realization. In *Proceedings of COLING 2002*, Morristown, NJ, USA, 2002.

[3] Hubert Haider. Downright down to the right. *Linguistik Aktuell*, 11:245–271, 1996.

[4] Karin Harbusch and Gerard Kempen. Clausal coordinate ellipsis in German: The TIGER Treebank as a source of evidence. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, 2007.

[5] D. Hindle and M. Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120, 1993.

[6] Ron M. Kaplan and John Maxwell. Constituent coordination in lexical-functional grammar. In *Proceedings of COLING 1989*, Budapest, Hungary, 1989.

[7] Sandra Kübler. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*, Borovets, Bulgaria, 2005.

[8] Sandra Kübler, Wolfgang Maier, Ines Rehbein, and Yannick Versley. How to compare treebanks. In *Proceedings of LREC 2008*, Marrakech, Morocco, 2008.

[9] Wolfgang Maier. Annotation schemes and their influence on parsing results. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, Sydney, Australia, 2006.

[10] Gereon Müller. On extraposition and successive cyclicity. In *Syntax: Critical Concepts in Linguistics*, volume III. Routledge, 2006.

[11] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications, Chicago, Illinois, 1994.

[12] Ines Rehbein and Josef van Genabith. Treebank annotation schemes and parser evaluation for German. In *Proceedings of EMNLP/CoNLL 2007*, Prague, Czech Republic, 2007.

[13] Ivan A. Sag, Gerald Gazdar, Thomas Wasow, and Steven Weisler. Coordination and how to distinguish categories. Technical report, CSLI-84-3. Center for the Study of Language and Information, Sumford, California, 1984.

[14] Anne Schiller, Simone Teufel, and Christine Thielen. Guidelines für das Tagging deutscher Textkorpora mit STTS. Technical report, Universität Stuttgart and Universität Tübingen, 1995.

[15] Mark Steedman. Dependency and coordination in the grammar of Dutch and English. *Language*, (61):523–568, 1985.

[16] Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Universität Tübingen, Germany, 2005.

MonaSearch – A Tool for Querying Linguistic Treebanks

Hendrik Maryns and Stephan Kepser*
Collaborative Research Centre 441
University of Tübingen, Germany
{hendrik,kepser}@sfs.uni-tuebingen.de

Abstract

MonaSearch is a new powerful query tool for linguistic treebanks. The query language of MonaSearch is monadic second-order logic, an extension of first-order logic capable of expressing probably all linguistically interesting queries. In order to process queries efficiently, they are compiled into tree automata. A treebank is queried by checking whether the automaton representing the query accepts the tree, for each tree. Experiments show that even complex queries can be executed very efficiently. The tree automaton toolkit MONA is used for the computation of the automata.

1 Introduction

A frequent critique to most available query tools is their poor expressive power: most of them only incorporate a subset of first-order logic as a query language. Furthermore, often the semantics of the query language is not clearly defined; it is only defined by the functionality of the program. The importance of the expressive power of the query language is a consequence of the sizes of the available treebanks nowadays: tree banks of several tens of thousands of trees are no exception. It is clearly impossible to browse these treebanks manually searching for linguistic phenomena. But a query tool that does not permit the user to specify the sought linguistic phenomenon quite precisely is not too helpful, either. If the user can

*The work presented here is part of the project A2 in the Collaborative Research Centre 441 "Linguistic Data Structures" at the University of Tübingen, funded by a grant of the German Research Council (DFG SFB 441). We would like to thank Michael Jakl, Uwe Mönnich and Fang Wei for interesting and fruitful discussions. We are particularly grateful to Michael for pointing us to the function `mgCheck` in the MONA gta library. We also thank the anonymous reviewers for their remarks.

only approximate the phenomenon he seeks, answer sets will be very big, often containing several hundreds to thousands of trees. Weeding through answer sets of this size is still cumbersome and not really fruitful. If the task is to gain small answer sets, then query languages have to be powerful.

As a query language offering the necessary expressive power, we present Monadic Second-Order logic (henceforth MSO). [Kepser, 2004] describes its power and how it can be exploited to query tree banks in linear time. This paper describes how these ideas were successfully implemented to obtain the most powerful query engine for treebanks to date.

MSO is an extension of first-order predicate logic by set variables. These variables can be quantified over, representing (finite) sets of nodes in a tree. MSO offers a firm expressive power, while at the same time the automaton model discussed below assures the evaluation time of an MSO query on a tree is linear in the size of the tree.

An example of its strength is the ability to express the transitive closure of any binary relation which is definable in the language. Since MSO is an extension of first-order logic, any first-order relation can be expressed in it, and thus also the transitive closure of any first-order relation.

Transitive closures appear quite often and very naturally in linguistics. Most often, the need for them has been met by introducing atomic formulas which directly express the transitive relation. For the ancestor relation, for example, an atomic formula $x \lhd^+ y$ is introduced, next to the more basic parent relation $x \lhd y$. The limitation of this approach is that those relations are hard-coded in the logic. There is no way to define new ones apart from extending the language. In the case of a corpus query tool, this would mean the extra relation has to be implemented by the programmer himself. In MSO, this can be done dynamically by the user. As it is hard to foresee the need for such relations, this feature can come in handy.

An example illustrating this, which might be regarded as a 'natural' query for MonaSearch, is the following: Consider sentences ending in a sequence of prepositional phrases like

> The dog buried the bone [PP behind the tree [PP in the garden [PP in front of the house [PP at the end of the street]]]].

where the PPs are all embedded by one another as indicated. This contrasts with a structure like in

> The dog buried the bone [PP with his paws] [PP under a stone] [PP behind the tree] [PP in the afternoon].

where each PP is an independent adjunct of the verb. Both examples can stepwise be extended to sequences of PPs of arbitrary length. The query for an arbitrary

number of embedded PPs is not definable in first-order logic, but can be defined in MSO. We can define the transitive closure of a PP dominating an NP as a new relation and state that the highest PP node and the most deeply embedded NP node stand in this new relation.

Another property of MSO which comes in nicely is its decidability over trees[1]. This assures that, when a query gives no results, this effectively means no tree in the tree bank satisfies the formula, rather than that no tree was found.

In this paper, we present MonaSearch, a query tool which implements MSO as a query language using automata techniques. This way we show that the theoretical considerations are indeed also practically valid. We compare run times of a set of linguistically motivated queries to show that MonaSearch is not only powerful, but also faster than its main competitors.

MonaSearch is publicly available at `http://tcl.sfs.uni-tuebingen.de/MonaSearch`.

2 Overview of the Querying Process

What makes MSO usable as a query language is the existence of an automaton model for this logic. The type of automata that correspond to MSO is bottom-up tree automata. For a general introduction to tree automata, we refer the reader to Comon et al. [2007]. The connection between MSO and tree automata is very strong. A set of trees is definable by an MSO formula if and only if there exists a tree automaton accepting this set. This equivalence is constructive: there is an algorithm that constructs an automaton from a given MSO formula.

Hence the general evaluation strategy of MonaSearch can be summarized as follows: 1. convert the query from the user into a tree automaton, 2. run the automaton on each tree in the tree bank. In addition, some auxiliary steps are necessary to smooth computation.

This strategy has the advantage that checking whether an automaton accepts a tree is very fast, viz. linear in the size of the tree.

3 MONA

The largest and most demanding subpart of this process is the development of a tree automata toolkit, a toolkit that compiles formulas into tree automata by performing standard operations on tree automata such as union, intersection, negation, and determinization. The most notable and successful is MONA [Klarlund and Møller,

[1] More precisely, over graphs with bounded tree width.

2001]. Although developed for hardware verification, MONA can be used to query treebanks.

The language of MONA is pure monadic second-order logic of two successors. Note that there is no way to extend this language. This has important consequences: Firstly, we are restricted to using *binary* trees only; secondly, it cannot handle node labels.

The main part of MONA is a compiler that compiles formulas in an idiosyncratic logical language into a specific variant of tree automata. The input is a file containing the formulas. The default output is some information on whether the formula is satisfiable or not, it is however possible to let it output a representation of a compiled automaton into a file. Additionally, MONA offers a small library with functions that allow one to load those precompiled automata from file and to check if an automaton accepts a binary coded tree.

The strategy for employing MONA to query treebanks consists of two parts: Since we don't want to burden the user with learning the complex MONA language, a query from the user in MSO is translated into a MONA formula, and this formula is compiled into a tree automaton by the main program. The library functions are then used to check each tree in the treebank for acceptance. The trees from the treebank have to be transformed in a suitable way described below such that they can be used as input to MONA's particular type of automata.

4 MonaSearch

In this section, we'll go into some more detail about the steps involved in a query and the preparatory tasks, and explain how they are handled in MonaSearch.

4.1 Precompilation of the Treebank

Linguistic treebanks are usually provided in some type of data exchange format. This format is typically unsuitable for direct querying. Plain text files are not suitable for efficient processing by their nature. Bouma and Kloosterman [2007] propose an efficient way of handling XML-encoded treebanks, but it requires the treebank to be in a specific format. E.g. the TüBa-D/Z [Telljohann et al., 2004] and TIGER [Brants et al., 2004] treebanks cannot be queried in that way. MonaSearch does not suffer from this restriction.

In the case of MonaSearch, there is a particular reason for a precompilation step. As mentioned before, MONA can only cope with *proper binary* trees. When translating trees from the treebank into MONA trees we consider proper trees only. Many treebanks contain more complex structures than proper trees. At the current

stage of the development of MonaSearch we simplify these structures as follows: 1. secondary relations are ignored, 2. disconnected subparts are integrated into the tree by introducing a new virtual root and connecting the disconnected parts to this super root and 3. crossing edges are ignored by taking only the order of the children as seen by the parent into account. In principle, these restrictions can be released: There exists a more general method that encodes tree-like structures into proper binary trees, as is illustrated in [Kepser, 2004].

As stated above, the precompilation of trees into formulas has to perform two tasks: 1. the trees, which are arbitrarily branching, must be transformed into binary trees and 2. the linguistic labels have to be taken care of.

For the transformation into binary trees, we employ the First-Child-Next-Sibling encoding, a rather standard technique. An arbitrary node x in the original tree is transformed into x' in the binary tree as follows:

- If x has any children, call its leftmost child y, then y' will become the left child of x'.
- If x has any right siblings, call the leftmost one z, then z' will become the right child of x'.

For example, the left tree in fig. 1, stemming from the TüBa-D/Z treebank, is transformed into the tree on the right. For real trees, the original tree can easily be recreated from this binary form; in fact, that is what the formula translation described later is implicitly doing. The simplifications mentioned before, however, are not retrievable from the binary form and are therefore not available for querying. Since we refer to trees using an identifier, we do not need to do this back-translation explicitly, we simply use the original tree for further processing.

Note how the disconnected punctuation node at the lower right corner in the original tree becomes the right child of the SIMPX node in the transformed tree; it is treated as if it was its sibling in the original tree.

As can be seen in the figure, edge labels are moved down to the node below it; thus, a node in the translated tree carries a set of labels. This is necessary either way, since even in the original tree, the leaves carry multiple labels. We'd like to mention that it is only a coincidence of the chosen annotation scheme and source tree that all and only the leaf nodes of the binary tree are labelled with words. This need not be the case in general.

Presently, labels for the category of a node, its function, the word at a leaf, its lemma or its morphological information occur, however, other labels can be treated the same way.

The preprocessing of the labels is still not enough to enable MONA to handle the binary trees. The trees it expects contain bit vectors as labels, which are closely

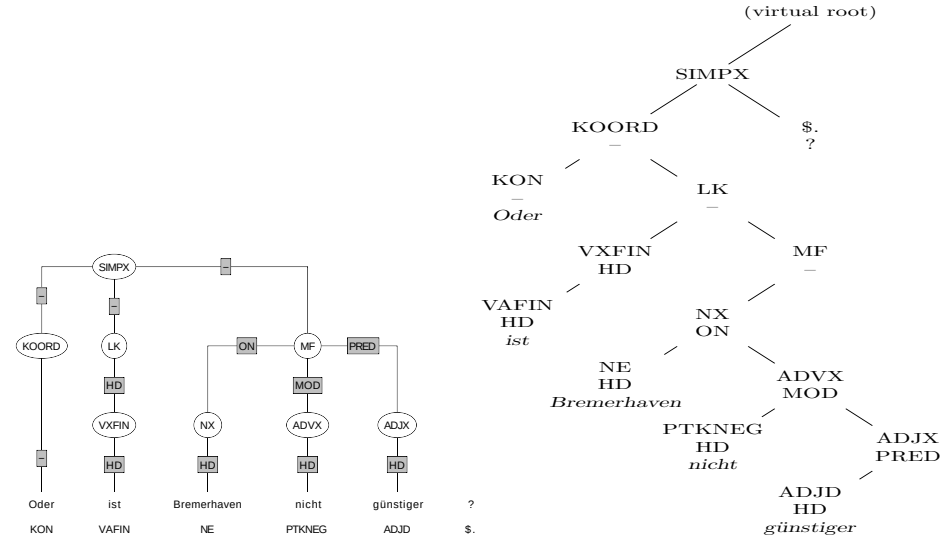


Figure 1: An example sentence from TüBa-D/Z and its binary coding.

related to the translation process. These bit vectors are, however, dependent on the specific query and are therefore generated while querying.

4.2 Querying

The detailed steps of a query are as follows:

1. getting a query from the user;
2. translating the query into a MONA formula on binary trees;
3. compiling the MONA formula into a MONA tree automaton;
4. for each tree from the precompiled treebank:
   1 preparing the tree for the query,
   2 running the automaton on the translated tree, noting whether it is accepted or not;
5. presenting the results to the user.

Steps 1 and 5 are handled in a graphical user interface.

For step 2 one has to keep in mind that the user formulates his query on the original treebank, but the trees that are used for querying are the binary coded trees. Hence it is necessary to translate the original query ϕ in such a way into a MONA formula ϕ' that the original query ϕ is true for an original tree t if and only if the translated formula ϕ' is true on the binary recoding t' of t.

This step also has the advantage that it makes us independent of the idiosyncratic MONA language. We can offer the user an easily understandable, semantic-

Relation	Formula	Translation
parenthood	$x \lhd y$	`ex1 z: x.0 = z & right_branch(z,y)`
precedence	$x \prec y$	`ex1 z: ( x = z | dom(z.0,x) ) & dom(z.1,y)`
dominance	$x \lhd^+ y$	`ex1 z: x.0 = z & dom(z,y)`

Table 1: MONA translation of the more involved base relations.

ally clear language that is almost the pure MSO logic. It also makes extending the query language with typical linguistic relations such as c-command relatively easy.

For most connectives, the MONA counterpart can be taken over directly. This is the case for all boolean connectives, quantification and some of the atomic relations. The main problem lies with the relations that express something about the shape of the tree, dominance and precedence being the most relevant. Their translation uses two auxiliary predicates on the binary tree, which can be defined in the MONA language:

- `dom`(x,y) expresses that x dominates y in the binary tree.
- `right_branch`(x,y) expresses that y lies at the branch of right children starting at x.

Note that both express the transitive closure of a first-order relation (parenthood and right child, respectively) and as such cannot be expressed in first-order logic.

Table 1 shows the translation of the parent and precedence relations on the original trees into a formula on binary trees in the language of MONA; see the user manual for the notation. Note that the dominance relation could equally well be expressed as the transitive closure of the parent relation. We include it since it turns out that a direct translation yields a simpler formula.

Finally, all types of linguistic labels are treated as free set variables. This yields a MONA formula ϕ'.

In step 3, the MONA formula is compiled into a tree automaton by a call to the MONA compiler. This implies that MONA has to be installed on the system in order for MonaSearch to work.

Step 4 comprises the actual querying of the treebank. The aim is to run the tree automaton on each binary coded tree.

Remember that MONA is not able to handle the preprocessed trees directly. The process of compiling a formula into an automaton translates variables into bit vectors. More precisely the label of each node in a tree processable by the automaton is a bit vector. The index of the vector states which variable is stored at that place. The value (0 or 1) indicates if the node is in the set of this variable. Since linguistic labels are treated as set variables, each binary tree must undergo this translation process. We impose some (arbitrary, but fixed) order on the labels. The length of the bit vector is the number of labels in the query. The index states

which particular label is referred to. Now consider a node in the binary tree and its original set L of labels. The new label is a bit vector. If the index refers to a label in L, the bit is set to 1, otherwise it is set to 0. As a consequence, the bit vectors in the translated tree only refer to labels occurring in the query.

After this translation step the input tree is finally ready for processing. The MONA library contains functions that load the precompiled tree automaton from file and apply it to a translated tree returning true or false depending on whether the tree automaton accepts the tree or not.

It remains to solve one small technicality. Remember that in the binarization process, a virtual root node was introduced to connect disconnected parts of the 'tree'. Care has to be taken to ignore this node when evaluating the query, in order not to skew the results. Here, fortune has smiled upon us: the particular form of tree automaton MONA uses happens to do just that when evaluating a simple tree: it ignores the root node and starts looking at its left child. This left child will necessarily be the first root node of the original tree, no further tweaking is needed.

4.3 User interface

MonaSearch offers MSO as a query language. The usual logical connectives are provided, along with first- and second-order quantification. For the atomic formulas, those relevant for relations of nodes in a tree are chosen, along with relations that express relations between nodes and node sets. Furthermore, (linguistic) labels of nodes can be queried. First-order variables are supposed to range over nodes, whereas second-order variables, the particular property of MSO, range over sets of nodes.

The user can compose these formulas stepwise in a bottom-up fashion in the graphical user interface, which acts like a scrap book of formulas that can be edited, connected using the logical connectives, and submitted for querying. Base formulas can be entered from menus.

We illustrate the query language by an example of a query. In most main sentences in German – as in English – the subject precedes the inflected verb. In yes/no-questions, this order is reversed. To find trees where the verb precedes the subject in the TüBa-D/Z treebank, the following query may be used:

$$\exists x, y, z(cat(x) = \text{SIMPX} \wedge cat(y) = \text{VXFIN} \wedge fct(z) = \text{ON} \wedge x \lhd^+ y \wedge x \lhd^+ z \wedge y \prec z)$$

"There exist a node with category SIMPX, a node with category VXFIN and a node with function ON; the first dominates the other two, and the second comes before the third." (*ON* codes the grammatical function *subject* in the TüBa-D/Z.) The tree in fig. 1 is a sample match for this query.

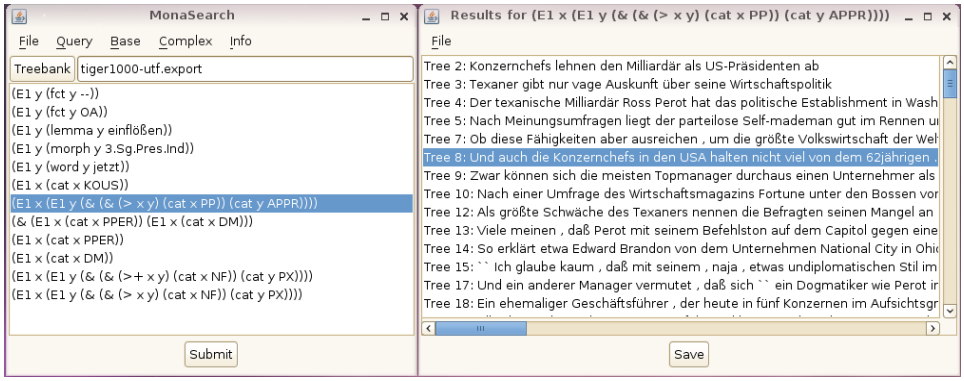


Figure 2: The MonaSearch GUI: compose and result window

5 Performance

In order to give an impression of the performance of MonaSearch we compare query times for a series of queries for three query tools: TIGERSearch [Lezius, 2000], fsq [Kepser, 2003], and MonaSearch. These tools were chosen because they can be used to query any treebank and already have rather high expressiveness. We tested their forces on the TIGER treebank and on TüBa-D/Z. The queries are ordered by complexity starting with simple ones.

1. An NP dominating an S dominating a PP.
2. An NP dominating an S dominating a PP and an NP, which do not dominate each other.
3. Sentences where the verb precedes the subject.
4. An NP *not* dominating an S which dominates a PP.
5. A PP dominating an NP which is part of a chain of embedded PPs (see introduction).

Table 2 shows that MonaSearch performs quite well. TIGERSearch may be faster on simple queries, but its deficit in expressive power makes it unsuitable for querying more advanced linguistic phenomena. In fsq, such phenomena can be queried, but MonaSearch is faster and even more powerful. Furthermore, fsq requires a certain skill in composing a query. Processing a query with quantifier depth 4 takes 23 seconds when the query is composed optimally. But it can also take 46 minutes, when the query is composed in a less skilled fashion. MonaSearch does not suffer from this problem. The processing time of logically equivalent queries is largely independent of the particular phrasing of the queries. This makes MonaSearch significantly easier to use.

Query	TIGERSearch		fsq		MonaSearch	
1	5.5	5.5	23	13.5	15	10
2	9	5.5	23	13.5	15	10
3	15	16	23	13.5	15	10
4	–	–	23	13.5	15	10
5	–	–	–	–	15	10

Query times in seconds. For each query tool, the left column contains the query time on the TIGER treebank, the right column the one on the TüBa-D/Z. A dash (–) indicates that the query language of the tool is not powerful enough to express the query.
System: Pentium D, 3.2GHz, 1GB Ram, OS: Linux openSUSE 10.3

Table 2: Query times of MonaSearch in comparison to other tools

Since all trees are evaluated separately, the evaluation time of a query is linear in the number of trees in the treebank. Effectively, a huge treebank with one million trees can be queried in about five minutes, i.e., quite fast.

6 Related Work

Alternative query tools and their performance have just been discussed in the previous section. The aim of this section is to explain how the current approach differs from the one in [Kepser, 2005]. Kepser [2005] explores a different strategy for using MONA to query linguistic treebanks. This different strategy proved unsuccessful. The reason for this being that trees were encoded as MONA automata featuring all linguistic labels of the treebank. As a result, the automata which encoded the trees got so big they could not be processed any more. The current approach does *not* try to code trees as automata.

The current approach avoids large label sets on trees by equipping each tree that is to be checked by an automaton with only those labels which are relevant to the current query, as described in step 4.1 in section 4.2. By avoiding these pitfalls the method presented in this paper is successful.

7 Conclusion and Future Work

In this paper, we presented MonaSearch, a powerful tool for querying linguistic treebanks. The two main features of MonaSearch are the very high expressive power of the query language and the high performance of the query engine. The query language is monadic second-order logic. This is – to our knowledge – the most expressive query language offered in any query tool useful for linguists. Mon-

aSearch is on the other hand also the fastest query system available for advanced queries. Thanks to MonaSearch's speed it is for the first time within reach to query automatically annotated treebanks with several million trees with non-trivial queries.

Future work has two directions. Firstly, usability will be enhanced by simplifying the syntax of the query language, extending the range of treebank input formats and enhancing the GUI. An important component would be a graphical treebank browser. Secondly, it would be worth investigating whether the restriction to proper trees can be overcome. As stated, there exists a theoretical solution for this problem. But the recoding of trees and formulas that is involved in this solution is very complicated. Formulas may get very large. It is therefore an open question whether these formulas can still be compiled into automata. But the potential advantage of coping with arbitrary tree-like structures should make it worthwhile to investigate the practicability of this solution.

References

Gosse Bouma and Geert Kloosterman. Mining syntactically annotated corpora with xquery. In *Proceedings of the Linguistic Annotation Workshop*, pages 17–24, Prague, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-1503`.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620, 2004.

Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2007. URL `http://tata.gforge.inria.fr/`.

Stephan Kepser. Finite Structure Query: A tool for querying syntactically annotated corpora. In Ann Copestake and Jan Hajič, editors, *Proceedings EACL 2003*, pages 179–186, 2003.

Stephan Kepser. Querying linguistic treebanks with monadic second-order logic in linear time. *Journal of Logic, Language, and Information*, 13:457–470, 2004.

Stephan Kepser. Using MONA for querying linguistic treebanks. In Chris Brew, Lee-Feng Chien, and Katrin Kirchhoff, editors, *Proceedings HLT/EMNLP 2005*, pages 555–563, 2005.

Nils Klarlund and Anders Møller. *MONA Version 1.4 User Manual*. BRICS, Department of Computer Science, University of Aarhus, January 2001. URL `http://www.brics.dk/mona/`.

Wolfgang Lezius. Tigersearch – ein suchwerkzeug für baumbanken. In Stephan Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*, Saarbrücken, 2000.

Heike Telljohann, Erhard W. Hinrichs, and Sandra Kübler. The TüBa-D/Z treebank – annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2229–2232, 2004.

Constructing a Valence Lexicon for a Treebank of German

Erhard W. Hinrichs, Heike Telljohann
University of Tübingen
Seminar für Sprachwissenschaft
E-mail: {eh, telljohann}@sfs.uni-tuebingen.de

1 Introduction

Treebanks allow for the creation of a valence lexicon per side effect. The TüBa-D/Z valence lexicon has been created in lockstep with the development of the TüBa-D/Z treebank as such. For each verb encountered in the treebank, the annotators created a lexical entry that records the valence frames of the verbs contained in the sentence, unless they are already contained in the valence lexicon as result of previous annotation. The TüBa-D/Z valence lexicon currently contains a total of 8013 frames for 4896 distinct verb lemmas. Since treebank annotation is still ongoing, the lexicon will continue to grow.

Such a lexicon has utility in its own right as a resource for lexicalized parsing and a variety of NLP applications. At the same time, the lexicon can serve as a source for aiding consistency of annotation and automatic detection of annotation errors.

2 Characteristics of the TüBa-D/Z Treebank

The TüBa-D/Z treebank of German is a linguistically annotated corpus based on data of the German newspaper 'die tageszeitung' (taz). Currently, it comprises approximately 36 000 sentences. Its scheme for the syntactic annotation is derived from the Verbmobil Treebank of spoken German (Hinrichs et al., 2000), which has been enhanced along various dimensions to accommodate the characteristics of written texts. The annotation scheme of the TüBa-D/Z treebank comprises four levels of syntactic annotation: the lexical level, the phrasal level, the level of topological fields, and the clausal level. The primary ordering principle of a clause is the inventory of topological fields, which characterize the word order regularities

among different clause types of German, and which are widely accepted among descriptive linguists of German (cf. (Drach, 1937; Höhle, 1986)). A set of node labels describes the syntactic categories (including topological fields and coordinations). The context-free backbone of phrase structure (i.e. proper trees without crossing branches) (Telljohann et al., 2004) is combined with edge labels specifying the grammatical function of the phrase in question as well as long-distance relations. For more details on the annotation scheme see Telljohann et al. (2006).

3 The Valence Lexicon

The development of the treebank is accompanied by the creation of a valence lexicon, which is growing with the size of the treebank. For each verb that is contained in the treebank, a lexical entry is created that consists of the infinitive form of the verb (lemma), the valence frame for the specific sentence, the sentence reference number, and the sentence itself in abbreviated form (marked by 'Bsp:' (example)). When the same verb occurs in another sentence with a different valence frame, this frame is added unless it is already contained in the valence lexicon. Maintaining a valence lexicon of this kind is motivated by the necessity of a reference list that ensures a consistent syntactic annotation of grammatical functions throughout all sentences in the treebank. All new entries of the valence lexicon are checked in a manual correction phase with special regard to previous entries of the respective verb lemma. In case of a detected error within a valence frame, all occurrences of the verb in the treebank are inspected. This strategy is labor intensive but decreases the error rate of human manual extraction of lexical information (cf. O'Donovan et al. (2005) for an automatic approach to extract subcategorization frames from Penn-II and Penn-III treebanks).

Example (1) shows an example entry of the valence lexicon for the verb *einsetzen*, which is polysemous and occurs with different valence frames in the treebank[1]:

(1)	einsetzen:	ON [einsetzen] OA	(R4-5603)
	=======	Bsp: Wir haben Computer eingesetzt	
		'We used the computer.'	
		ON [einsetzen] OA FOPP (für, gegen)	(R4-3126)
		Bsp: Wir setzen uns für eine Feuerpause ein	
		'We supported a cease fire.'	

[1]The codes surrounded by parentheses refer to the number of the current/new treebank release (R4/N5) followed by the sentence number where this particular valence frame occurs (typically the first occurrence).

Bsp: Gegen den Widerstand setzt der Senat Polizeiknüppel ein (R4-27058)
'Against the resistance the senate used billy clubs.'

ON [einsetzen] (R4-2903)
Bsp: Schneefall hatte eingesetzt
'Snowfall had set in.'

ON [einsetzen] OA PRED (R4-17034)
Bsp: Gourmetköche setzen sie als Garnitur ein
'Gourmet cooks used it as garnish.'

ON [einsetzen] OD OA (N5-37382)
Bsp: Man setzt den Pflanzen neue Gene ein
'One inserts new genes into the plants.'

Table 1 summarizes the grammatical functions used in the valence lexicon:

Label	**Description**
ON	nominative object (incl. subject clauses)
OG	genitive object
OD	dative object
OA	accusative object
OS	sentential object
OPP	obligatory prepositional object
FOPP	facultative prepositional object
OADVP	adverbial object
OADJP	adjectival object
PRED	predicate
OV	verbal object

Table 1: Grammatical functions in the valence lexicon

The inventory of grammatical function labels used in the valence lexicon in Table 1 coincides with the edge labels used in the syntactic annotation of the TüBa-D/Z. Thus, the categories used in the TüBa-D/Z valence lexicon correspond directly to syntax. This distinguishes this valence lexicon from other valence lexica whose labels abstract away from surface syntax and encode primarily semantic information, such as PropBank (see Palmer et al. (2005)), FrameNet (see Baker et al. (1998)), and the Prague valency lexicon PDT-VALLEX (see Hajič et al. (2003)).

By default, valence frames record verb usage in the active voice. Valence frames of passive and infinitive clauses are added to the valence lexicon as specific

entries only if a verb occurs exclusively in a passive or in an infinitive sentence in the treebank (e.g. the valence frame "ON [anfordern] (PASSIV)" for the sentence *Die Schablone wird angefordert* ('The template is being ordered.') or "[angeln] (INFINITIV)" for the clause ... *in der Themse zu angeln* ('... to fish in the Thames.'). As soon as the same verb is encountered in an active or finite sentence, the passive or infinitive frame is replaced by the active or infinitive counterpart. This ensures that verbs that are only used in passive or infinitive constructions in the treebank can be easily identified.

Any developer of a valence lexicon needs to determine how much information should be contained in individual valence frames. In particular, one needs to decide in advance whether valence frames should include only (obligatory) complements of the verb or whether (optional) adjuncts should also be incorporated. The TüBa-D/Z valence lexicon steers a middle ground with respect to these distinctions. In addition to obligatory complements, it includes those adjuncts that are characteristic of individual verbs. Such adjuncts typically take the form of PPs headed by a particular prepositional head, or they take the form of predicate modifiers. The former class is categorized by the label FOPP (for: facultative prepositional object; cf. Table 1), and the latter by the label PRED (cf. Table 1). On the other hand, adjuncts that are generally admissible for verbal predicates such as temporal or locative modifiers are not included in any valence frames. In the TüBa-D/Z treebank such modifiers receive the labels V-MOD (if the adjunct unambiguously modifies a verb) or MOD (for cases where the scope of the modifier cannot be uniquely determined and thus needs to remain underspecified).

The inclusion of prepositional or predicative adjuncts obviously leads to more fine-grained valence entries. This has at least two advantages for treebank development: (i) it supports a higher degree of consistency in the annotation of modifiers, in particular regarding the three labels FOPP, V-MOD, and MOD, which are particularly difficult for annotators to assign, (ii) it aids in the disambiguation of polysemous verbs since different verb senses often correlate with the presence or absence of verb-specific prepositional or predicative adjuncts.

4 Quantitative Analyses of the Valence Lexicon

The TüBa-D/Z valence lexicon currently contains a total of 8013 frames for 4896 distinct verb lemmas. These numbers are snapshot values documenting the current state of work. In order to estimate the lexicon's coverage, we counted the number of new verb lemmas and new frames found in ranges of 5000 sentences.

Figure 1 shows the accession rates for new verb lemmas, new frames, and the combinations of both (the distribution of 7743 frames for 4758 verb lemmas) in the

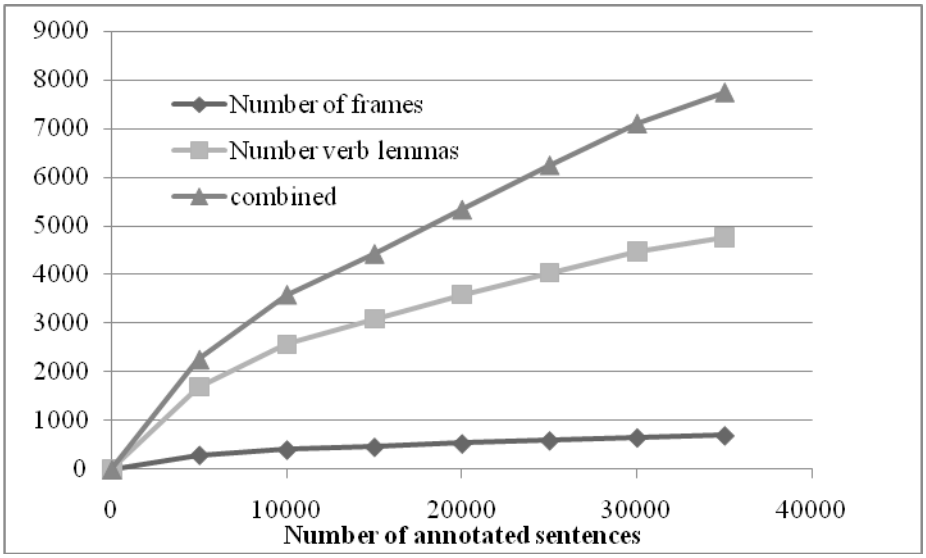


Figure 1: Accession rates for frames, verb lemmas, and their combinations

ranges up to sentence number 35 000.

For the first 5000 sentences the accession rate is 33,9% for verb lemmas and 6,0% for frames. These values drop down to 5,9% (verb lemmas) and 0,9% (frames) for the sentence number range 30 000 to 35 000. For the next range of 5000 sentences, an increase of approximately 250 new verbs (ca. 5%) can be expected. The accession rate for frames is about one sixth of the one for verb lemmas. None of the curves already indicates saturation.

Figure 2 illustrates the distribution of valence frames over sentence number range for the 15 verb lemmas with the highest number of valence frames (cf. Table 2). The stacked columns show that the highest increase of new frames occurs in the range (r) up to 5000 sentences followed by the range up to 10 000 sentences.

The subsequent sections present additional quantitative analyses, which have been performed on the valence lexicon.

4.1 Number of Distinct Valence Frames

Currently, the valence lexicon contains 717 distinct valence frames. The frequency of occurrence for a specific valence frame ranges from 2243 (ON OA) down to one. 488 different valence frames occur only once in the valence lexicon, 67 frames are encountered twice, and 36 frames are found three times.

Figure 3 shows a diagram of the thirty highest ranking valence frames. The

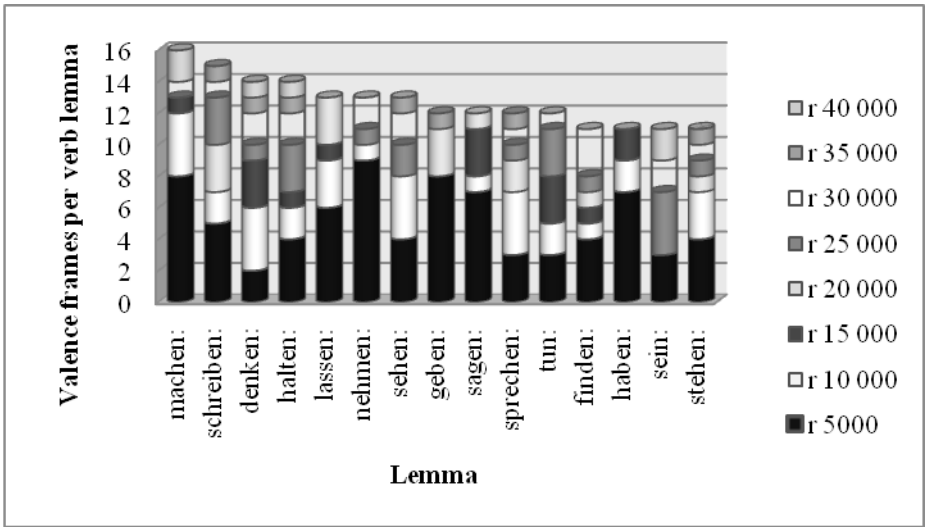


Figure 2: Distribution of valence frames over sentence number range for individual verb lemmas

relative frequency of frames reflects general as well as domain-specific tendencies. Among the six most frequently occurring frames are transitive verbs with accusative objects (rank 1), intransitive verbs (rank 2), verbs with sentential complements (rank 4), ditransitive verbs (rank 5), and transitive verbs with dative objects (rank 6). The fact that transitive frames occur much more frequently than intransitive frames can be attributed to the fact that most intransitive verbs also allow for transitive frames, thus boosting the relative frequency of the latter.

Perhaps the most surprising result among the top most frequent frames concerns passive frames (rank 3). Recall that passive frames "survive" in the valence lexicon only for those verbs for which no corresponding active frame is encountered in the treebank (cf. section 3). The high frequency of such purely passive uses of verbs is most likely due to the genre of newspaper texts that the TüBa-D/Z is based on. The fact that verbs with sentential complements are fourth-ranked may also be attributed to such a genre effect.

Notice also that valence frames are sensitive to the unmarked word order of complements and adjuncts in the topological field MF[2]. Thus, ON OD OA and ON OA OD constitute separate frames for ditransitive verbs. If one adds up these two frequency counts, then the two ditransitive frames jointly outnumber the frame for sentential complement taking verbs.

[2]For the unmarked word order in the field MF see Engel (1988), pp. 320ff.

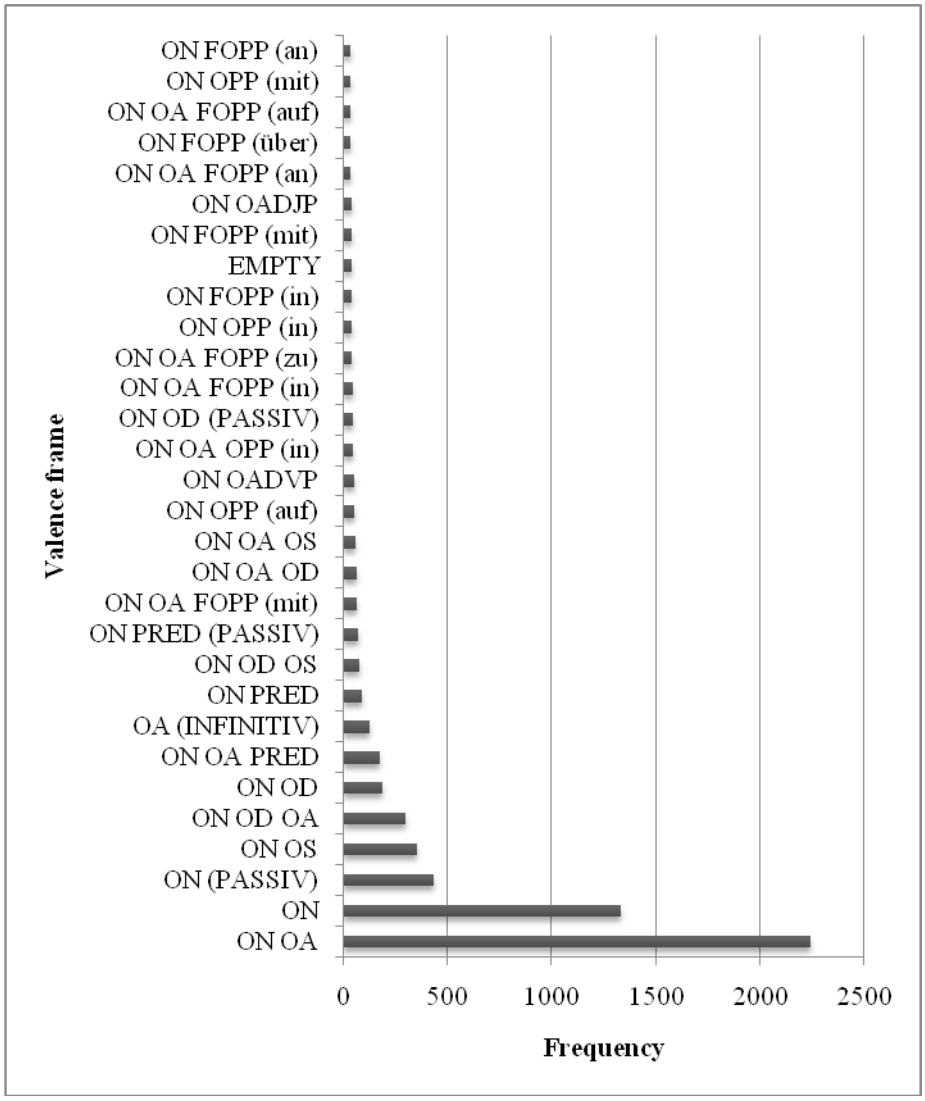


Figure 3: Top 30 list of valence frames

Verb lemma	Valence frames per verb lemma	Frequency count
machen	16	1
schreiben	15	1
denken, halten	14	2
lassen, nehmen, sehen	13	3
geben, sagen, sprechen, tun	12	4
finden, haben, sein, stehen	11	4
entscheiden . . . wissen	10	9
bleiben . . . verpflichten	9	6
bekommen . . . ziehen	8	15
anfangen . . . zahlen	7	25
abstimmen . . . wünschen	6	33
anbieten . . . zwingen	5	85
abfahren . . . zustimmen	4	146
abgeben . . . zweifeln	3	347
abbrechen . . . zutreffen	2	921
aalen . . . zwitschern	1	3294

Table 2: Valence frame count and frequency count

It is important to keep in mind that a corpus — or in this case a treebank — is always finite in size. Therefore, the relative frequency of frames is a reflection of what actually occurs in the particular corpus or treebank. Nevertheless, the results shown in Figure 3 show that a treebank of the size of the TüBa-D/Z reliably yields general tendencies in the distribution of valence frames. What has to be left to future research is a more precise profiling of general and domain-specific effects. In other words, how would a treebank of the same language that is based on a different genre, say literary texts or spoken language, differ from the results reported here for the TüBa-D/Z?

4.2 Number of Valence Frames for Each Verb

The analysis of the valence frame count value per verb results in the frequency distribution illustrated in Table 2.

The results in Table 2 show that 67,3% of all verb lemmas (3294 out of 4896 distinct verbs) occur in only one valence frame and 86,1% (3294+921 out of 4896 distinct verbs) realize at most two distinct valence frames in the treebank data. At the other end of the spectrum, there is a small number of verbs that exhibit up to

maximally 16 distinct frames. Most these verbs are semantically light such as *tun* ('do') and *machen* ('make') or are verbs that have both a main verb and an auxiliary usage such as *lassen* ('let'), *haben* ('have'), and *sein* ('be').

As in the case of the relative frequency of individual valence frames (cf. section 4.1), the number of distinct frames per verb in the valence lexicon is, of course, a direct reflection of the underlying corpus. In other words, a given lemma may well have additional valence frames, but these may just happen not to occur in the data. It is therefore an interesting question whether the number of distinct valence frames is in any way correlated with the number of occurrences in the corpus. This is the subject of the next section.

4.3 Lemma Frequency in the Treebank and its Correlation with the Number of Valence Frames per Verb

In order to be able to correlate the number of valence frames per verb found in the valence lexicon (cf. Table 2) with the number of occurrences of the corresponding lemmas in the treebank, a verb lemma list has been extracted from the treebank data.

Table 3 shows the top 20 lemmas in descending order together with the corresponding frame count values. The maximum lemma frequency is 10 009 for the lemma *sein* ('be') followed by *werden* ('will') , *haben* ('have'), *können* ('can'), *sollen* ('shall'), *müssen* ('must'), and *wollen* ('want'). All of them belong to the classes of auxiliary and modal verbs. The top ranking main verb is the lemma *geben* ('give') with a frequency of 1021. The maximum valence frame count is 16 for the lemma *machen* ('make'; lemma frequency 801).

The correlation analysis is based on a total of 4838 matches between the valence lexicon and the lemma list. Mismatches between both lists facilitate error detection and correction, for both the valence lexicon and the treebank.

The degree of correlation between lemma frequency and valence frame count can best be illustrated by plotting the two data series for each lemma/verb in a diagram.

In order to be able to compare the lemma frequency and valence frame count values, both data series have been scaled to a maximum value of 100 in Figure 4 (LF for lemma frequency and VFC for valence frame count).

Figure 4 indicates that valence frame count values in general jump erratically up and down. There is a weak correlation between the lemma count and the average valence frame count as can be seen by the linear trend line drawn for the valence frame count curve.

Figure 4 invites the following conclusions: 1. Since there is a weak correlation between lemma count and valence frame count, the number of distinct valence

Lemma	Lemma frequency	Valence frame count per verb
sein	10009	11
werden	6545	7
haben	5766	11
können	2164	6
sollen	1418	6
müssen	1373	5
wollen	1294	8
geben	1021	12
sagen	922	12
machen	801	16
kommen	668	10
lassen	626	13
gehen	562	10
stehen	475	11
sehen	462	13
bleiben	409	9
dürfen	379	5
heißen	364	10
wissen	364	10
finden	361	11

Table 3: Top 20 correlation of lemma frequency and valence frame count

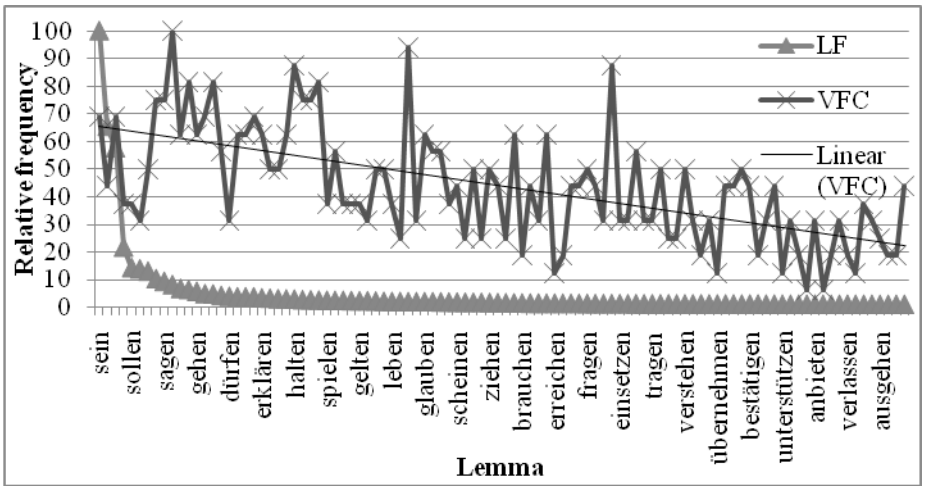


Figure 4: Top 100 correlation of lemma frequency and valence frame count

frames found in the treebank is at least partially determined by the number of occurrences of the lemma, 2. The fact that this correlation is only a weak shows that the number of valence frames per verb is largely dependent on the inherent properties of the verb itself, in particular its range of lexical meanings with their corresponding syntactic realizations.

5 Conclusion and Future Work

In this paper we have described the TüBa-D/Z valence lexicon that has been constructed in lockstep with the TüBa-D/Z treebank of German. Here we have concentrated on a quantitative analysis of the lexicon and emphasized the utility of such a lexicon for maintaining consistency of annotation.

In future research we plan to use the TüBa-D/Z valence lexicon not just for the treebank itself, but to integrate it into other resources under development for German. A particularly suitable resource for such an integration is GermaNet (Hamp and Feldweg, 1997), the German version of the English WordNet. Such a combined resource of the TüBa-D/Z valence lexicon and GermaNet would have a number of advantages: (1) if verb senses are matched with valence frames, this can clarify the intended sense of the verb, (2) it allows an empirical verification of the relationship between the correlation of distinct valence frames and sense distinctions.

References

Baker, Collin F., Fillmore, Charles J. and Lowe, John B. 1998. The Berkeley FrameNet project. In *Proceedings of the COLING-ACL*. Montreal, Canada.

Drach, Erich. 1937. *Grundgedanken der deutschen Satzlehre*. Frankfurt/Main: Diesterweg.

Engel, Ulrich. 1988. *Deutsche Grammatik*. Heidelberg: Groos.

Hajič, Jan, Panevová, Jarmila, Urešová, Zdeňka, Bémová, Alevtina, Kolářová, Veronika and Pajas, Petr. 2003. PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation. In J. Nivre und E. Hinrichs (eds.), *Mathematical Modeling in Physics, Engineering and Cognitive Sciences, 9, Proceedings of The Second Workshop on Treebanks and Linguistic Theories* (pp. 57-68). Växjö, Sweden.

Hamp, Birgit and Feldweg, Helmut. 1997. GermaNet - a Lexical-Semantic Net for German. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. Madrid.

Hinrichs, Erhard W., Bartels, Julia, Kawata, Yasuhiro, Kordoni, Valia and Telljohann, Heike. 2000 The Tübingen Treebanks for Spoken German, Englisch, and Japanese. In W. Wahlster (ed.), *Verbmobil: Foundations of Speech-to-Speech Translation* (pp. 550-574). Berlin: Springer.

Höhle, Tilman. 1986. Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder. In Akten des Siebten Internationalen Germanistenkongresses 1985 (pp. 329-340). Göttingen, Germany.

O'Donovan, Ruth, Burke, Michael, Cahill, Aoife, van Genabith, Josef and Way, Andy. 2005. Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II and Penn-III Treebanks. In *Computational Linguistics 31 (3)* (pp. 329-366).

Palmer, Martha, Kingsbury, Paul and Gildea, Daniel. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. In *Computational Linguistics 31 (1)* (pp. 71-106).

Telljohann, Heike, Hinrichs, Erhard W. and Kübler, Sandra. 2004. The TüBa-D/Z Treebank: Annotating German with a Context-Free Backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)* (pp. 2229-2232). Lisbon, Portugal.

Telljohann, Heike, Hinrichs, Erhard W., Kübler, Sandra and Zinsmeister, Heike. 2006. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.

A Quechua-Spanish Parallel Treebank

Annette Rios, Anne Göhring and Martin Volk
University of Zurich
Institute of Computational Linguistics
{ozli,Anne_Goehring}@access.uzh.ch, volk@cl.uzh.ch

1 Introduction

Most treebank work in the past has focused on European and Asian languages. The Wikipedia Treebank page lists treebanks (or treebank projects) for about 20 modern European languages (ranging from Basque to Swedish), five Asian languages (Chinese, Japanese, Hindi, Korean, Thai), two ancient languages (Greek and Latin), plus Arabic and Hebrew.

Almost no treebanking work has been done on African or American indigenous languages.[1] In the past we have explored parallel treebanks for English, German and Swedish [7]. Now we would like to explore to what extent our tools and guidelines will work when we include a very different language, Quechua, for which only few NLP resources exist. Since Quechua is spoken in Latin America, Spanish as parallel language is a natural choice.

We have first compiled a parallel corpus Quechua - Spanish. We have then stepwise analyzed and annotated the Quechua and the Spanish texts. For Spanish we have used the treebanking guidelines developed by [8]. As for Quechua there were no such guidelines so that we had to experiment with finding the appropriate grammar formalism and develop our own guidelines.

In this paper we describe the characteristics of Quechua and our steps towards its morphological and syntactic annotation. We argue for Role and Reference Grammar as a suitable grammar formalism. We briefly describe how we annotated the parallel Spanish texts and demonstrate how we plan to align the Quechua with the Spanish trees.

[1] a notable exception is the work by [5]

2 Our Quechua-Spanish Corpus

Quechua is a group of closely related languages, spoken by about 8 million people in Peru, Bolivia, Ecuador, Southern Colombia and in the North of Argentina. The Quechuan Languages are divided into two subgroups, QI and QII. Quechua I is the more archaic group of dialects, spoken in Central Peru. The internal diversity between these dialects is very high, mutual intelligibility not always given. It's very likely that the origin of the Quechuan Languages lies in this area [3].

fQuechua II itself consists of three subgroups, QIIA, spoken in Northern Peru; QIIB, spoken in Ecuador and Colombia and QIIC, spoken in Southern Peru, Bolivia, and Argentina[2]. In this project, the main focus lies on the dialects of the QIIC group, and within these, especially on Cuzco and Ayacucho Quechua. The reason why QIIC was chosen for this project is very simple: For QIIC, and particularly for Cuzco and Ayacucho Quechua, there are not only by far the most linguistic descriptions at hand, but there are also more bilingual texts available than for any other dialect.

There are a lot of bilingual texts in Quechua and Spanish on the web, ranging from political texts over news to poetry and even literature. Besides these electronic texts, there are also some Quechua-Spanish printed texts and translated books, for example *Don Quijote* by Miguel de Cervantes and *Le Petit Prince* by Antoine de Saint-Exupéry. We have chosen the following texts for this project:

- the *declaration of human rights*, which is available in various Quechua dialects and contains about 100 sentences.
- some information texts and the FAQ from the website of the Peruvian *Defensoría del Pueblo*[3], which all together contain about 100 sentences.

Spanish is an official language of twenty-one countries spoken by 320 million people all around the world. Despite its geographical extension and the great regional and national diversity, it is still considered to be one language. Yet European Spanish is no longer the exclusive model for modern standard Spanish and differences can be found in pronunciation, vocabulary, and even in syntax.

The chosen text genres both influence vocabulary and sentence length. The Spanish texts of our corpus contain many juridical expressions and scarcely present any Latin American or Peruvian characteristics. The numerous adjectives, enumerations, sentence coordinations and subordinations tend to lengthen the sentences;

[2]The letters A-C stand for the linguistic distance to QI, so QIIA is the most akin to QI, whereas QIIC is the most divergent group respective to QI

[3]The Defensoría del Pueblo is an institution that makes sure the state complies with its responsibilities for its citizens and that should also prevent the state from violating the rights of citizens.

on the other side, the short titles lower the average number of tokens per sentence to about 20.

3 Building the Quechua Treebank

3.1 Morphology

Quechua is a strongly agglutinative, suffixing language. For the morphological analyzer developed within this project we considered more than 130 suffixes. It makes sense to build Quechua syntax trees not only on whole words, but on their morphemes.[4] In order to do so, we had to develop a morphology tool that would automatically segment the words into morphemes. The challenge within this task is suffix order: Each local variety has its own preferred suffix order [3, 4], and variation in this order is not only allowed, but can sometimes even lead to a change in meaning. The solution to this problem is to group the suffixes according to their relative position in the verb. Generally, the suffixes follow this scheme:

Table 1: Suffix Order

Nominal Root	Derivation	Possession	Case		Ambivalent Suffixes
Verbal Root	Derivation	Aspect/Tense	Person	Modality	Ambivalent Suffixes

There are many ambivalent roots, that can take either verbal or nominal morphology without modification. Additionally, there are a couple of very productive nominalizing and verbalizing suffixes that can change a nominal root into a verbal one, and vice versa. Some of these suffixes can combine with each other:[5]

(1) *chinka -y.* — loss/lose -1.Sg.Poss — "My loss."

chinka -ni. — loss/lose -1.Sg.Subj — "I lose."

[4] For instance in cases without argument NPs, where subject (and object) are expressed only by verbal suffixes.

[5] Abbreviations used:

Abl	Ablative	Acc	Accusative	Ag	Nomen Agentis
AUX	Auxiliary	CLM	Clause Linkage Marker	Dat	Dative
Dem	Demonstrative Pronoun	DE	Direct Evidence	DS	Different Subject
Excl	Exclusive	Fact	Factitive	Gen	Genitive
IF	Illocutionary Force	Incl	Inclusive	Inf	Infinitive
Loc	Locative	NPst	Neutral Past	NRoot	Nominal Root
NS	Nominalizing Suffix	NUC	Nucleus	Obl	Obligation, Purpose
Perf	Perfect	Pl	Plural	PiP	Pl.Incl.Poss
Poss	Possessive Suffix	PRO	Proform	Rflx	Reflexive
Sg	Singular	Sim	Similarity	SS	Same Subject
Subj	Subject	TNS	Tense	VRoot	Verbal Root
VS	Verbalizing Suffix				

(2) *kachi* *-cha* *-sqa* *wiña* *-y* *-cha* *-ku* *-y*
salt -Fact(VS) -Perf(NS) grow -Inf(NS) -Fact(VS) -Rflx -Inf(NS)
"salted,salty" "to perpetuate oneself"

We used Xerox Finite State Tools (xfst) to build our morphological analyzer [1]. First of all, we split up Quechua Suffixes into five classes (table 2). Three out of these five classes needed further refinement, namely the N–>N, V–>V and the ambivalent suffixes.

Table 2: Suffix Classes

1	nominalizing suffixes	V –> N
2	verbalizing suffixes	N –> V
3	nominal derivational suffixes	N –> N [6]
4	verbal derivational suffixes	V –> V
5	ambivalent suffixes	N/V –> N/V

The nominal derivational suffixes (N–>N) were divided into 6, the verbal derivational suffixes (V->V) into 7 slots according to their relative position in the word. Some of these slots are iterable, i.e. more than one suffix out of a group is possible, while others are not. If more than one suffix of a given slot is present in the wordform, the relative order of these suffixes is variable, reflecting the differences between the various local varieties of the language.

The class of the ambivalent suffixes contains suffixes that are attached to nominal or verbal wordforms, without changing their part of speech. The position of these suffixes is at the end of the suffix sequence, their relative order is more or less fixed, dialects show some minor variation.

3.2 Syntax Trees

In a first attempt, we tried to build the Quechua syntax trees using phrase structures. However, Quechua poses some severe problems for this approach, and so we looked for a more appropriate grammar formalism. With respect to its complex morphological structure Quechua is similar to languages like Finnish and Estonian. Treebanks for these languages have also avoided constituent structure trees. The Estonian Arborest Treebank, for example, is based on constraint grammar which is a special type of dependency structure. [2] mention that non-finite clausal constructions pose special problems for their formalism. They solve the issue by leaving certain dependencies between subclauses underspecified. We propose that

[6] contains also possessive suffixes and case markers

RRG (Role and Reference Grammar) as described by Van Valin [9] is best suited to account for the characteristics of Quechua, including the non-finite clausal constructions, for the following reasons:

3.2.1 NP vs. VP

There is no clear-cut differentiation into NPs and VPs. Embedded clauses always contain non-finite, nominalized verbforms. These nominalized verbs are clearly nominal, they carry nominal morphology (possessive and case markers), but they also have subjects and objects, and so are clearly predicative elements. How are these forms to be treated in a constituent tree? They are no verbal phrases, but whole clauses, with their own arguments, and so they would have to be treated as clauses (S) with a nominal head. However, it seems rather unusual to have a sentence node without a finite verb in a constituent tree. A similar problem arises from the fact that the copula for 3rd person singular may be dropped, resulting in a sentence with no finite verb.
In RRG on the other hand, the predicative element PRED is not restricted to a single part of speech, in fact, any wordform can be predicative. Hence there is no problem having a CLAUSE with a noun as predicative element. The case markers of the nominal clause can be treated as Clause Linkage Markers (CLM), according to [9].

3.2.2 Headless Relative Clauses

A special form of nominalization are the so-called headless relative clauses. Such relative clauses without external head are quite common in Quechua. Consider the following example:

(3)

..ley	*-man*	*-hina*	*derecho*	*-nchik*	*-pa*	*contra*	*-n*	*-pi*	*ruwa*	*-q*
law	-Dat	-Sim	right	-1.PiP	-Gen	against	-3.Sg.Poss	-Loc	do,make	-Ag
-kuna	*-manta*	*-m*	*waqa*	*-y*	*-cha*	*-sqa*	*ka*	*-na*	*-nchik.*	
-Pl	-Abl	-DE	shelter	-Inf	-Fact	-Perf	be	-Obl	-1.PiP	

"..so that, according to the law, we are protected from those who act against our rights."[7]

[*derechonchikpa contranpi ruwaqkunamantam*] - "from [the ones] who act against our rights" is a relative clause without head. The verbal root *ruwa-* bears the nominalizing suffix *-q* (*Nomen Agentis*), followed by the plural marker *-kuna* and the case suffix *-manta*, which are clearly nominal. If there was an external

[7] Article 8 of the Declaration of Human Rights: "Everyone has the right to an effective remedy by the competent national tribunals for acts violating the fundamental rights granted him by the constitution or by law."

head, plural and case markers would be attached to the head instead.[8] So *ruwaq* is clearly a predicative element, in this case without arguments, but it could as well have. Nevertheless, its outer node cannot be a clause, since it bears a plural suffix[9], which leads to the conclusion that the whole clause has to be considered as a nominal element. The solution in RRG is to assume a NP which contains a CLAUSE with a nominal predicative element (*ruwaq*). This approach follows exactly what [9] proposes for Lakhota nominal relative clauses.[10]

3.2.3 Switch Reference

Yet another special case is Switch Reference (Clause Chaining). Consider the following sentence from the text *Llaqtaman sayapakuq -Beatriz Merino* on the website of the *Defensoría del Pueblo*.

(4)

Chay	*-ta*	*-m*	*aypa*	*-rqa*	*-ø,*	*San*	*Marcos*	*Hatun*	*Yacha*	*-y*
Dem	-Acc	-DE	achieve	-NPst	-3.Sg.Subj	San	Marcos	big	know	-Inf
Wasi	*-manta*	*"Mariano*	*Ignacio*	*Prado"*	*beca*	*-yuq*	*ka*	*-spa.*		
house	-Abl	Mariano	Ignacio	Prado	stipend	-Poss	be	-SS		

"When [she] had achieved this, [she] obtained a 'Mariano Ignacio Prado' stipend from the San Marcos University."

[*Chaytam ayparqa*] is the main clause with a finite verb, whereas [*San Marcos Yachay Wasimanta Mariano Ignacio Prado becayuq kaspa*] is the chained, nominalized clause. The problem with phrase structures is now, besides the issue whether the embedded clause has to be treated as VP or S (or even NP), the nexus type itself. To treat the embedded clause as coordinate is not accurate, yet the embedded clause shares evidentiality and tense with the main clause and it has no finite verb. But [... *becayuq kaspa*] is not subordinated either: There is no morpheme indicating the semantic relation to the main clause, nor is the embedded clause some kind of clausal object. Rather, the two clauses describe a sequence of events. In RRG, there is a third nexus type, cosubordination, that allows to represent the clauses as two clauses on their own, but sharing evidentiality (IF), see figure 1.[11]

So finally, RRG was chosen over phrase structures, although also within this framework, there is one major issue, namely the double-marking nature of Quechua. Van Valin and La Polla [9] assume that every language is either predominantly

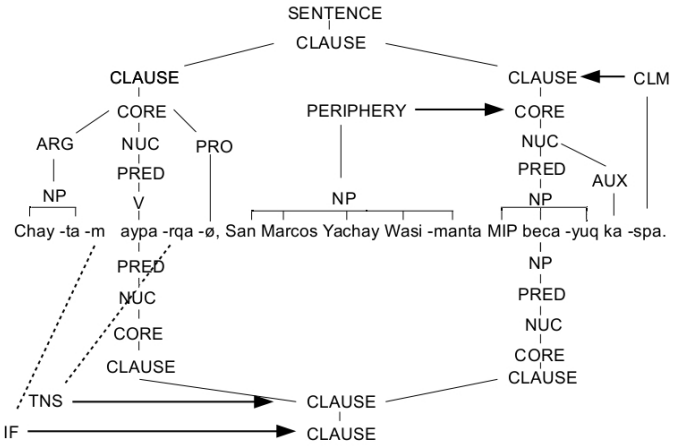
[8] e.g. with *runa*, "person": [*derechonchikpa contranpi ruwaq*] *runa**kunamanta*** - "from the persons who act against our rights"

[9] Of course the case suffix *-manta* is also a nominal suffix, but case markers can be treated as Clause Linkage Marker, see 3.2.1

[10] Van Valin's Lakhota relative clauses are internally headed, as opposed to the Quechua example, that has no head at all. But the structure is the same: an NP containing a (relative) clause.

[11] glosses see sentence 4 in 3.2.3

Figure 1: Quechua Sentence with RRG (simplified)



head- or dependent-marking. The difference is that in head-marking languages the verbal affixes are attached as PRO (Proforms) to the core, whereas the (unmarked!) NPs are considered to be outside the core and so are attached to the clause node. Considering that in Quechua the finite verb always bears person suffixes for its arguments,[12] but a sentence without argument NPs is possible, it seems more plausible to treat Quechua as a head-marking language, at least for local persons. Yet we decided to make a compromise for 3rd person objects: On account of the fact that these are never cross-referenced on the verb we chose to attach them to the CORE instead of assuming a zero-morpheme.[13] In a sentence with 1st or 2nd person object, which are cross-referenced on the verb, Quechua would be treated as head-marking, so that the core would contain the suffixes, but not the argument NPs.

We built the RRG syntax trees with the tool Annotate-3.6, which was developed to build phrase trees. RRG in fact has three levels of annotation: constituent projection, focus structure, and operator projection. This results in a three-dimensional structure. It's impossible to build such trees within Annotate-3.6, which only provides nodes, edges and secondary edges. So we decided to leave out focus structure. We then built the constituent projection without major problems, using

[12] except for 3rd person objects, which are always zero-marked

[13] As opposed to the zero-morpheme for 3rd person subject in the example sentence, which is absolutely plausible, because the 3rd person singular marker *-n* is optional after the tense suffixes *-rqa* and *-sqa*.

nodes and some labeled edges.[14] The operators were connected directly to their corresponding nodes via edges annotated with the appropriate labels. Because of the restriction in Annotate-3.6 that a word (in our case suffixes) can only be attached to one node, there were cases where secondary edges had to be used to represent operators, namely for suffixes expressing person and future tense, respectively modality all in one.

4 Building the Spanish Treebank

To syntactically annotate our Spanish corpus we used a modified version of the AnCora tagsets.[15] AnCora has three levels of annotation: a morphological, a syntactic and a semantic level. In this project, we focused on the manual syntactic annotation and kept the semantic level for future work.

On the morphological level AnCora distinguishes between the part of speech (PoS) and categories such as gender, number, case, person, time, and mode. We have simplified its morphological tagset by keeping the PoS and cutting the morphological information. Instead of having 280 different labels, we reduced the set to 33 PoS tags; then we added a label for foreign words, so that the number of PoS tags is now 34.

On the syntactic level, the AnCora corpora are annotated with constituents and functions. We reduced the constituents so that they are similar to the set of phrase constituents used in the German Negra Corpus. One of our main principles is to keep the annotation simple for the annotators. To facilitate and speed up their job they should annotate as flat as possible without losing information; in a second step we will automatically deepen the structure to obtain the same tree as if following the AnCora guidelines (similar to the deepening we have used in previous projects [6]). We thus discarded some intermediate constituent nodes, typically the nodes just under the phrases. There is another difference on the token level: AnCora has single and multiword tokens: a person's first and last name are analyzed as one token as in *(Miguel_Indurain)*. We leave the tokens separate and group them under a constituent node MPN (multi-token proper name). Other cases of multiword tokens are adverbial or conjunctive expressions like *ni_siquiera* resp. *a_pesar_de*. Again, we defined other special constituent labels to gather these complex expressions together: MTC (multi-token conjunction) and MTP (multi-token preposition). The resulting constituent tagset has 19 labels.

As for the syntactic functions, we decided to keep all the function labels in a first phase; depending on the results of this experiment, we might drop some of the

[14] PERIPHERY, PRO, AUX and ARG

[15] freely available from http://clic.ub.edu/ancora

more complex and unused labels. The function labels serve to tag only the edges under a sentence constituent S; they correspond to traditional syntactic functions (subject, object, attribute, etc.) and discourse and modality elements.

Spanish is a pro-drop language, the subject pronoun, unless emphatically used, is normally omitted. In this case, the sentence structure simply lacks a subject function. When the subject of a coordinated or subordinated sentence is elliptical, a secondary edge connects the existing subject to that sentence's constituent node.

To solve the problem of multiword tokens, as we did with the multi-token constituent nodes MPN, MTC and MTP, we defined a function SVC (support verb construction) to label the edges of the elements belonging to a light verb expression like *tener en cuenta*.

5 Aligning Quechua to Spanish

We used the Stockholm TreeAligner[16] for the alignment between the trees. Aligning Quechua to Spanish is a difficult task since the syntactic structures of the two languages differ a lot:

- Spanish uses prepositions, whereas Quechua almost exclusively uses suffixes.
- Different grammatical properties are encoded: for example, Spanish marks definiteness of NPs via articles, whereas Quechua doesn't mark definiteness, but instead marks a NP as being the topic or focus of the clause.
- Quechua uses evidential suffixes to mark the source of knowledge for each proposition, Spanish lacks a comparable category.

Often, the texts are not translated literally; the meaning is given, but with different structures. Even worse, corresponding information is often split up between various sentences. For these reasons, it is difficult to find exact alignments. Often, only fuzzy alignments were possible, if any alignment at all. Figure 2 shows an example of a Spanish sentence aligned to a Quechua subordinate clause, red lines meaning fuzzy, green lines meaning exact alignments[17] (translation see below).

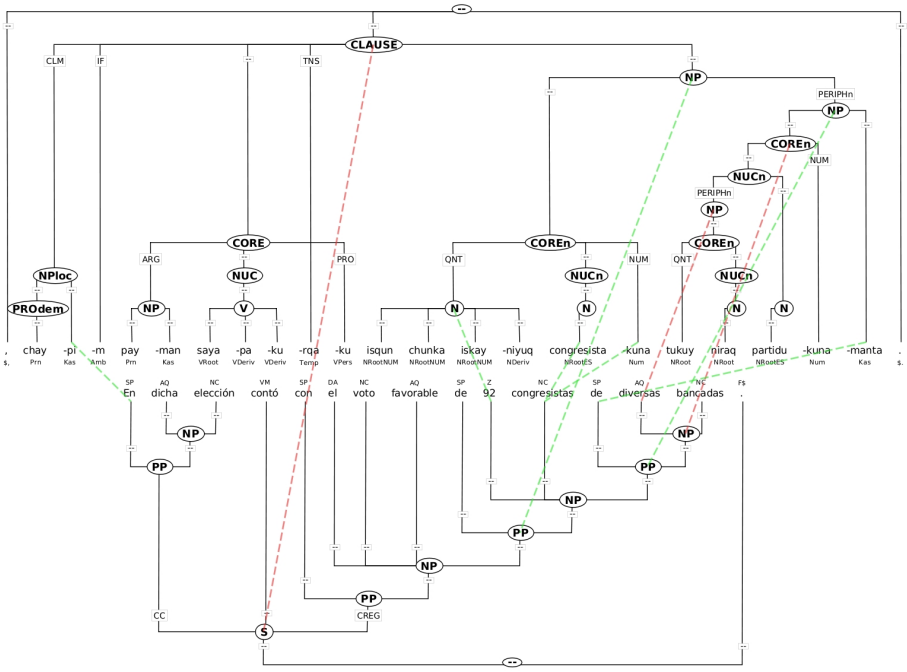
The two sentences differ in the way they express the same proposition. The literal translations would be:

- Spanish: "In the mentioned election, [she] counted with the favourable votes of 92 parliamentarians from diverse factions."

[16]The TreeAligner is available free of charge from: http://dev.ling.su.se/treealigner

[17]The Quechua main clause was cut out in this figure, for lack of space.

Figure 2: Alignment



- Quechua: “In this, [for] her stood up 92 parliamentarians from all sorts (colors) of parties.”

As you can see in Figure 2, we chose to align suffixes with prepositions when they convey the same meaning and would be good translations in other contexts too, as for example Spanish *en* and Quechua *-pi* (locative). On the other hand, *en dicha elección* - “In the mentioned election” and the corresponding Quechua part *chaypim* - “In this” could not be aligned because *en dicha elección* wouldn’t be a translation for *chaypim* in other contexts. Additionally, since the Quechua clause lacks the information conveyed by the PP *en dicha elección*, the sentence-to-clause alignment is only fuzzy (red lines). Contrary to this, the Spanish PP *de diversas bancadas* and the Quechua NP *tukuy niraq partidukunamanta* were aligned as exact matches: the internal structure is different, but the meaning conveyed is the same.

As a result of splitting up the Quechua words to their roots and suffixes, there are many multiple alignments from one Spanish word to more than one Quechua

token. For instance the Spanish word *congresistas* corresponds exactly to the Quechua *congresista* and *-kuna*[18]. In such cases, we allowed for exact multiple alignments (green lines).

6 Conclusions

We have found more bilingual texts Quechua-Spanish than we had expected. Since Quechua is a strongly agglutinative language we have decided to annotate the Quechua treebank on morphemes rather than words. This allows us to link morpho-syntactic information precisely to its source. In order to split the Quechua words into morphemes we have built a morphological analyzer based on standard finite state technology.

We realized that building phrase structure trees over Quechua sentences does not capture the characteristics of the language. We have therefore chosen Role and Reference Grammar. By using nodes, edges and secondary edges in our annotation tool we were able to represent the most important aspects of Role and Reference syntax for Quechua sentences. In order to represent all three dimensions of this formalism we will need to adapt our annotation and alignment tools.

So far, we have built the syntax structures for Quechua completely manually (after the automatic morpheme splitting). In the future we will integrate Part-of-Speech tagging and shallow parsing into the process. We will also work with alignment suggestions once we have reached a sufficiently large parallel treebank for training.

References

[1] Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Publications, 2003.

[2] Eckhard Bick, Heli Uibo, and Kaili Müürisep. Arborest - a Growing Treebank of Estonian. In Henrik Holmboe, editor, *Nordisk Sprogteknologi. Nordic Language Technology. Å rbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*. Museum Tusculanums Forlag, Copenhagen, 2004.

[3] Rodolfo Cerrón-Palomino. *Lingüística Quechua*. Centro de Estudios Regionales Andinos Bartolomé de Las Casas (CBC), 2. edition, 2003.

[4] Antonio G. Cusihuamán. *Gramática Quechua: Cuzco-Collao*. Gramáticas referenciales de la lengua quechua. Ministerio de Educación, 1976.

[18] *kuna* is the suffix indicating plural, just as Spanish *-s*.

[5] C. Monson, Ariadna Font Llitjos, Roberto Aranovich, Lori Levin, Ralf Brown, Eric Peterson, Jaime Carbonell, and Alon Lavie. Building nlp systems for two resource-scarce indigenous languages: Mapudungun and quechua. In *Proc. of the Fifth International Conference on Language Resources and Evaluation (LREC)*, 2006.

[6] Yvonne Samuelsson and Martin Volk. Automatic node insertion for treebank deepening. In *Proc. of 3rd Workshop on Treebanks and Linguistic Theories*, Tübingen, December 2004.

[7] Yvonne Samuelsson and Martin Volk. Phrase alignment in parallel treebanks. In Jan Hajic and Joakim Nivre, editors, *Proc. of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 91–102, Prague, December 2006.

[8] Mariona Taulé, M. Antònia Martí, and Marta Recasens. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008.

[9] Robert D. Van Valin Jr. and Randy J. La Polla. *Syntax - Structure, Meaning and Function*. Cambridge Textbooks in Linguistics. Cambridge University Press, 1997.

A Data-Driven Dependency Parser for Romanian

Mihaela Călăcean and Joakim Nivre
Uppsala University
Department of Linguistics and Philology
E-mail: mcalacean@sdl.com
joakim.nivre@lingfil.uu.se

Abstract

We present the first data-driven dependency parser for Romanian, which has been developed using the MaltParser system and trained and evaluated on a dependency treebank for Romanian developed within the RORIC-LING project. The parser achieves a labeled attachment score of 88.6% (unlabeled 92.0%) when evaluated on held-out data from the treebank. We present a partial error analysis, focusing on accuracy for different parts of speech and dependencies of different length.

1 Introduction

Data-driven methods for syntactic parsing currently enjoy widespread popularity, mainly because of the relative ease and efficiency with which parsers can be developed, provided that appropriate data sets for training are available. In recent years, considerable attention has been given to data-driven parsers that use representations based on the notion of dependency, where syntactic structure is represented by a hierarchy of syntactic relations between words. Dependency representations have been claimed to be especially well suited for languages with free or flexible word order, and research has shown that data-driven dependency parsers that are both efficient and accurate can be developed with fairly modest amounts of data. The potential of data-driven dependency parsing has been demonstrated on a large scale in the CoNLL shared tasks in 2006 and 2007, where parsers have been trained and evaluated using data from some twenty languages [2, 12].

In this paper, we add to the increasing literature in this field by presenting what we believe to be the first data-driven dependency parser for Romanian. The parser has been trained and evaluated on a treebank developed within the RORIC-LING project, using the freely available MaltParser system, and achieves a labeled

attachment score of 88.6% on held-out test data. This seemingly impressive result, given a training set of less than 35,000 tokens, is partly explained by the data selection procedure for the treebank, where only short and simple sentences were included, and by the fact that gold standard part-of-speech tags were used in the input at testing time. Nevertheless, the results are promising enough to motivate further work to extend the capability of the parser.

The remainder of the paper is structured as follows. Section 2 introduces the treebank that has been used for training and evaluation, and section 3 describes the MaltParser system. Section 4 contains the experimental results, and section 5 gives conclusions and suggestions for future research.

2 A Dependency Treebank for Romanian

The treebank developed in the RORIC-LING[1] project consists of 36,150 tokens (punctuation excluded) and comprises newspapers articles, mostly on political and administrative subjects. It contains 4,042 sentences, having a mean sentence length of 8.94 tokens per sentence. The type/token ratio of 0.245 indicates a rather frequent repetition of certain types. The texts were chosen so as to offer a representative sample of modern written standard Romanian. However, texts including complex ambiguities were avoided as much as possible and removed from the treebank.[2]

The strong tradition of applying the Dependency Grammar (DG) formalism in linguistic research on Romanian and in teaching prescriptive grammar in Romanian schools justifies the choice of annotation style. A description of a DG approach to syntactic analysis with special reference to Romanian can be found in [5]. The texts were annotated with part-of-speech (POS) tags and information about the dependency relations (annotation of the head and the dependent) and dependency labels. All the dependency graphs for the sentences in the treebank are connected, projective, rooted, acyclic and any node has at most one head.

The annotation was performed completely manually by a Romanian linguist, using only the graphical interface tool Dependency Grammar Annotator (DGA).[3] Since there was only one annotator, the POS tags and dependency types were relatively coherently used throughout the whole material. The annotated texts are

[1] RORIC-LING is the Romanian part of BALRIC-LING, an Information Society Technologies project aimed at raising awareness on Human Language Technologies and possible scientific and industrial applications of linguistic resources in Bulgarian and Romanian. More at http://www.larflast.bas.bg/balric/index.html.

[2] The information regarding the choice of texts from the treebank was included in the Romanian version of the RORIC-LING Bulletin available at http://www.phobos.ro/roric/Ro/qa16.html.

[3] Downloadable at http://www.phobos.ro/roric/DGA/dga.html.

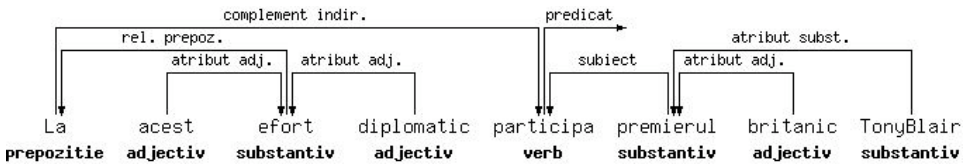


Figure 1: The sentence *La acest efort diplomatic participa premierul britanic Tony Blair* [Lit. 'The British prime-minister Tony Blair is part of this diplomatic effort'] annotated by DGA.

automatically saved by DGA in the XML format, using the ASCII character encoding, thus explaining the absence of the five Romanian diacritics (ă, â, î, ş and ţ) from the treebank.

The POS tagset used for annotating the treebank is relatively small and rather simple considering the morphosyntactic richness of Romanian. The dependency types set is exhaustive for the texts the treebank consists of. The problematic cases were most of the time avoided. All idiomatic and complex group structures were decomposed into simple elements, sometimes removing difficult structures. The representation of elliptical constructions in Romanian was considered unnecessary, mostly because the written language tends to eliminate them. Consequently, a shallow syntactic annotation was adopted. All complex-compound sentences (including coordinated sentences) were split into simple sentences, therefore there are no subordinate clauses in the treebank. Dates and measure phrases do not receive any kind of special annotation. There are no punctuation marks (except for a few hyphens, brackets and slashes in words like, e.g., *e-mail* or *NATO/Rusia [ro]*). Proper names consisting of two or more elements were collapsed into one lexical element (E.g., *Tony Blair* becomes a lexical unit *TonyBlair*). Figure 1 shows a sentence from the treebank developed in the RORIC-LING project.

In order to evaluate the accuracy of manual annotation, we randomly selected 3% of the total number of sentences (i.e., 122 sentences) and manually corrected all the errors, thus creating a gold-standard material. The original annotation and the gold standard were compared, and the agreement was found to be 98.9% for the dependency annotation, as measured by the labeled attachment score (LAS) of the original annotation with respect to the new gold standard. This result shows that the quality of the material is satisfactory. The errors occurring in the treebank include incorrect dependency labels and head identification problems.

Even if the annotation of the material was performed by a single annotator, inconsistencies still occur, especially within the annotation scheme. Four of the twenty POS tags and one dependency type appear only in the first 6% of the material, reducing significantly the POS tagset for the rest of the material. For instance,

verbs and adjectives in participle form are annotated as such only in the first part of the material. On the other hand, the definite article POS tag is present only in the last 90% of the material. In order to eliminate these inconsistencies, we have mapped the POS tagset used in the first part to the one used in the larger part of the treebank.

The treebank was intended first of all as a step towards linguistic resources in Romanian and, secondly, for training and evaluating statistical dependency parsers for Romanian. As far as we know, this is the first time the treebank has been used for the second purpose.

3 MaltParser

MaltParser [13] is a language-independent system for data-driven dependency parsing, based on a transition-based parsing model [10]. More precisely, the approach is based on four essential components:

- A transition-based deterministic algorithm for building labeled projective dependency graphs in linear time [11].
- History-based feature models for predicting the next parser action at nondeterministic choice points [1, 7, 15].
- Discriminative classifiers for mapping histories to parser actions [6, 19].
- Pseudo-projective parsing for recovering non-projective structures [14].

Given that all dependency structures in the Romanian treebank are strictly projective, the pseudo-projective parsing technique has not been used in the experiments and is not further described in this paper.

3.1 Parsing Algorithm

The parser uses the deterministic algorithm for labeled dependency parsing first proposed in [11]. The algorithm builds a labeled dependency graph in one left-to-right pass over the input, using a stack to store partially processed tokens and adding arcs using four elementary actions (where TOP is the token on top of the stack and NEXT is the next token):

- **Shift**: Push NEXT onto the stack.
- **Reduce**: Pop the stack.
- **Right-Arc(*r*)**: Add an arc labeled *r* from TOP to NEXT; push NEXT onto the stack.

	POS	LEX	DEP
TOP	*	*	*
TOP+1	*		
TOP−1	+		
TOP−2	+		
NEXT	*	*	
NEXT+1	*	*	
NEXT+2	*		
NEXT+3	*		
NEXT−1	+		
HEAD(TOP)	+	*	
LDEP(TOP)	+		*
RDEP(TOP)	+		*
LDEP(NEXT)	+		*
RSIB(LDEP(TOP))		+	

Table 1: History-based features used in the experiments, divided into default features (*) and features added specifically for Romanian (+). Symbols: TOP = token on top of stack; NEXT = next input token; HEAD(w) = head of w; LDEP(w) = leftmost dependent of w; RDEP(w) = leftmost dependent of w; RSIB(w) = next right sibling of w. Positive subscripts on TOP indicate relative position in the stack; other subscripts indicate relative position in the input string (negative = left, positive = right).

- **Left-Arc**(r): Add an arc labeled r from NEXT to TOP; pop the stack.

Parser actions are predicted using a history-based feature model (section 3.2) and SVM classifiers (section 3.3).

3.2 History-Based Feature Models

History-based parsing models rely on features of the derivation history to predict the next parser action [1]. The features used are all symbolic and defined in terms of three different token attributes:

- POS = part of speech
- LEX = word form
- DEP = dependency type

Features of the type DEP have a special status in that they are extracted during parsing from the partially built dependency graph and are updated dynamically during parsing. The other two feature types (LEX, POS) are given as part of the input to the parser and remain static during the processing of a sentence. The use of POS features presupposes that the input has been preprocessed by a part-of-speech tagger but for the experiments reported below we use the gold standard tags from the treebank.

Starting from the default feature model in MaltParser, we have performed forward feature selection to adapt the parser for Romanian. Table 1 shows the features used in the experiments, where rows identify tokens in a parser state, columns identify token attributes, and cells identify features defined by the row token and the column attribute. Cells marked * correspond to features in MaltParser's default model, while cells marked + indicate features added during feature selection for Romanian. Results for the different feature models are presented in section 4.

3.3 Discriminative Classifiers

We use support vector machines [18] to predict the next parser action from a feature vector representing the history. More specifically, we use LIBSVM [3] with a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$ and the built-in one-versus-all strategy for multi-class classification. Symbolic features are converted to numerical features using the standard technique of binarization, and we use MaltParser's default settings for all parameters associated with the learning algorithms.

4 Experiments

The total data set was split into a training set, a development set and an evaluation test set. The development set was used for preliminary testing and tuning the parsing models and the feature model specific for Romanian. The test set of 404 sentences, unseen during the development phase, was used for a final test run, training on 90% of the data (approximatixely 32,500 sentences). All results presented in this paper are on the final test set.

On the whole, there were several modifications brought to the original treebank data. First of all, the training and testing files used in the experiments were converted from XML into the CoNLL data format.[4] Secondly, some errors and inconsistencies detected in the material were fixed, as described in section 2. Thirdly, the presence of blanks and punctuation characters inside the strings of the original

[4]More information about the CoNLL format for dependency treebanks can be found at http://nextens.uvt.nl/depparse-wiki/DataFormat.

Feature Model	LAS	UAS
Default	87.9 (±2.1)	91.5 (±1.8)
Optimized	88.6 (±2.0)	92.0 (±1.7)

Table 2: Parsing accuracy measured in labeled and unlabeled attachment score (LAS and UAS).

POS tags and dependency labels led to the decision to map the original symbols to shorter, more compact but absolutely equivalent symbols. Finally, the special root node of the dependency graphs for sentences in the treebank was modified from being an extra dummy word at the end of each sentence to being an extra dummy word at the beginning of the sentence. Since the special root node does not correspond to any real token of the sentence, we have considered it a pure notational convention, having no theoretical or practical consequences.

As indicated in section 3, we have used two feature models in the experiments: the default model and a feature model optimized for Romanian. The results of the experiments are presented in Table 2, showing how the two feature models influence the parsing accuracy, evaluated with respect to labeled and unlabeled attachment score (LAS and UAS). The labeled attachment score represents the percentage of tokens that have been assigned both the correct head and the correct dependency label, while the unlabeled score considers only whether the head has been assigned correctly. Scores are reported with a 95% confidence interval, indicating that the differences between the default model and the optimized models are not statistically significant.

The scores for Romanian are very similar to those obtained using MaltParser 0.4 in the CoNLL 2007 Shared Task for configurational languages like English, Italian and Catalan [4]. Languages with rich morphology and flexible word order like, for instance, Czech have lower results. Given that Romanian can be considered a language characterized by flexible word order, plus a relatively rich morphology, these results are rather unexpected, especially given the limited amount of data available for training. However, as described in section 2, the texts were strictly selected, and complex syntactic structures were eliminated or simplified. This clearly facilitates the parsing task and explains the seemingly high accuracy, compared to results obtained for other languages and treebanks.

Table 3 presents the accuracy for different parts of speech, considering the percentage of tokens grouped under a certain part of speech for which both the head and the dependency relation to the head are predicted correctly. Predicative verbs have almost one hundred percent accuracy, most probably due to the fact that the treebank contains only simple sentences with only one verb acting as a predicate,

Part of Speech	LAS
Adjectives	95.1
Adverbs	90.8
Coordinating conjunctions	65.2
Nouns	89.1
Predicative verbs	99.5
Prepositions	69.2
Pronouns	89.1

Table 3: Parsing accuracy for different parts of speech

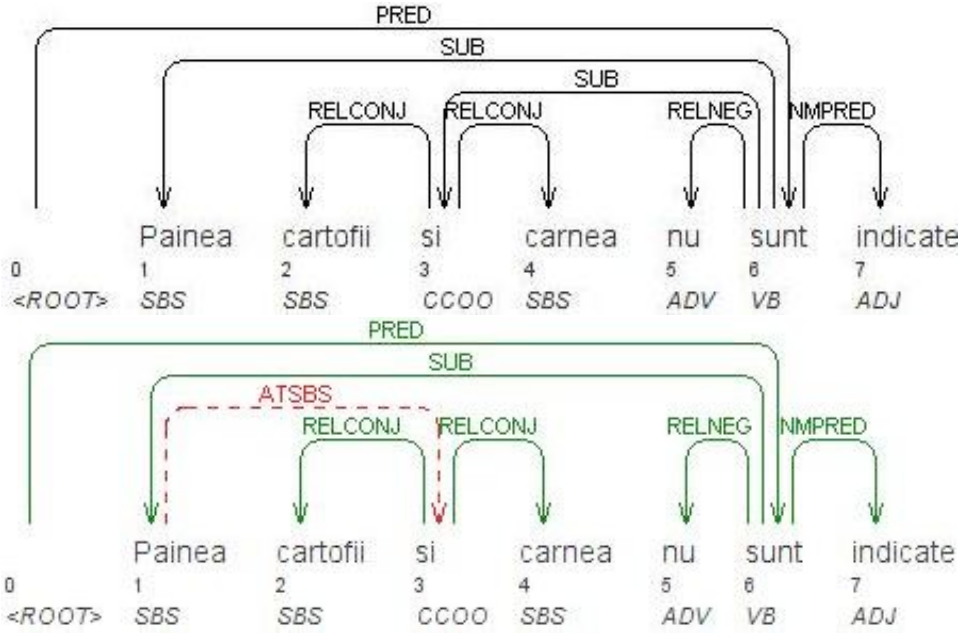


Figure 2: Gold standard (top) and predicted graph (bottom) for the sentence *Painea carnea si cartofii nu sunt indicate* [Lit. 'Bread, meat and potatoes are not advisable']. Correct dependencies are represented by solid arrows, incorrect dependencies by dashed arrows.

Length	P	R
ROOT	99.5	99.5
1	95.9	96.3
2	92.3	89.7
3–6	83.6	81.9
7–	60.8	71.3

Table 4: Labeled precision (P) and recall (R) for dependencies of different lengths; ROOT = dependents of the special root node.

also the root of the sentence. At the other end, prepositions and coordinating conjunctions have the lowest accuracy, being the hardest to attach correctly.

Figure 2 exemplifies the kind of errors the parser produces regarding coordinating conjunctions. In the gold standard annotation for the sentence *Painea carnea si cartofii nu sunt indicate*, the coordinating conjunction *si* ('and') is the head of two nouns: *cartofii* ('the potatoes') and *carnea* ('the meat'), while the conjunction itself is a dependent of the main verb, acting as a subject for the predicate *sunt* ('are') of the sentence. However, as the figure shows, the parser fails to predict this structure and instead analyzes the coordinate structure *carnea si cartofii* ('meat and potatoes') as a modifier of the noun *Painea* ('bread').

Table 4 shows the precision and recall for dependencies of different lengths (with dependents of the special root node in a separate category ROOT). The precision is the percentage of predicted dependencies of a certain length that are actually correct; the recall is the percentage of true dependencies of a certain length that are correctly predicted by the parser. As expected, both precision and recall decrease as dependencies get longer, a pattern that is well attested for other languages and treebanks [10]. However, the drop is less drastic than for many other data sets, with labeled precision and recall remaining above 80 for dependencies up to length 6, a result that can again probably be explained by the relative simplicity and homogeneity of the sentences in the treebank.

5 Conclusion

We have presented the first empirical results on parsing Romanian using the treebank developed in the RORIC-LING project. Using the freely available MaltParser system with a feature model adapted for Romanian, we achieve a labeled attachment score of 88.6% and an unlabeled attachment score of 92.0% on held-out test data from the treebank, using gold standard part-of-speech tags in the input to the parser.

The empirical results look very promising, but it must be remembered that the data in the treebank has been selected in such a way that only short and simple sentences have been included. This means that the parsing accuracy reported, while perfectly valid for the type of sentences included in the treebank, is not representative for the harder task of parsing unrestricted text in Romanian. The most important direction for future research is therefore to investigate different techniques for extending the capability of the parser to more complex sentences. Besides the obvious approach of simply annotating a larger training set, including sentences of arbitrary complexity, it is worth considering whether the existing parser can be used to speed up the process, using a weakly supervised approach involving active learning [17, 16] and/or self-training [8, 9]. In order to parse unrestricted text, a part-of-speech tagger for Romanian must also be developed.

References

[1] Ezra Black, Frederick Jelinek, John D. Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, pages 31–37, 1992.

[2] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164, 2006.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[4] Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, 2007.

[5] Florentina Hristea and Marius Popescu. A dependency grammar approach to syntactic analysis with special reference to Romanian. In Florentina Hristea and Marius Popescu, editors, *Building Awareness in Language Technology*. University of Bucharest Publishing House, 2003.

[6] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pages 63–69, 2002.

[7] David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 276–283, 1995.

[8] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

[9] David McClosky, Eugene Charniak, and Mark Johnson. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 561–568, 2008.

[10] Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, 2007.

[11] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, 2003.

[12] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932, 2007.

[13] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 2007.

[14] Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106, 2005.

[15] Adwait Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10, 1997.

[16] Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 406–414, 2002.

[17] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 406–414, 1999.

[18] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[19] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206, 2003.

Automatic Annotation of Morpho-Syntactic Dependencies in a Modern Hebrew Treebank

Noemie Guthmann Yuval Krymolowski
Adi Milea
Technion – Israel Institute of Technology

Yoad Winter
Technion and Utrecht University

Introduction

Morpho-syntactic dependencies between sentence constituents are an inseparable part of syntactic analysis, in particular in Semitic languages. In those languages, because of the relatively free order of certain constituents, morpho-syntactic agreement features are sometimes the main clue for computational parsing models (Tsarfaty and Sima'an, 2007; Tsarfaty and Sima'an, 2008). However, despite their centrality for syntactic analysis, morpho-syntactic dependencies have so far not been annotated in Hebrew resources. In particular, such dependencies were not annotated in early versions of the Modern Hebrew Treebank (MHT, Sima'an et al. (2001)), which to date is the only publicly available resource with syntactic analyses for Modern Hebrew. By developing a method for automatically adding dependency annotations to a Modern Hebrew treebank, the MHT project has aimed to contribute to treebank development for Semitic languages as well as for other languages.

This paper describes the development and implementation of the morpho-syntactic dependency scheme used in the MHT project. We concentrate on *mother-daughter dependencies*, in which the morphological features of one or more daughter nodes affect the morphological features or syntactic analysis of the mother node. The annotation scheme for such dependencies is based on familiar but non-trivial assumptions about grammatical rules for Modern Hebrew. These rules are used for two purposes. The first purpose is to annotate the mother-daughter dependencies between nodes in the treebank. The second purpose is to use the generated dependencies for annotating morpho-syntactic features of compound constituents. The rules are described in XML format and implemented using Python scripts. The

scripts were run on a recent version of the MHT to produce dependency annotations in the MHT2, the most recent version of the treebank. A sample of the annotated dependencies in MHT2 was manually evaluated, which showed high accuracy of the automatic scheme. Errors detected mostly resulted from errors in the original syntactic annotation of the MHT, without the dependency annotations. Thus, the development of the dependency scheme and its automatic implementation also proved helpful in improving the quality of the manual annotation.

Similarly to the earlier versions of MHT, also one of the major syntactic resources for Arabic, the Penn Arabic Treebank (ATB, Maamouri et al. (2004)) does not include manual annotations of morpho-syntactic dependencies. Mother-daughter dependencies are also missing from another important Arabic resource - the Prague Arabic Dependency Treebank (PADT, Smrž et al. (2008)). The annotation schemes used for the MHT and the ATB are designed in close correspondence to the phrase structure grammar underlying the English Penn Treebank (Marcus et al., 1994). Because of this similarity between the MHT and the ATB, and since the rules for morpho-syntactic dependencies are very similar in Modern Hebrew and Standard Arabic, we expect our methodology for automatic annotation to be generally applicable to other "Penn-compatible" syntactic resources like the ATB. This expectation is further supported by cross-linguistic relations between NLP works on Hebrew and Arabic dealing with POS-tagging (Bar-Haim et al., 2007; Mansour et al., 2007). Resources for other Semitic languages like Amharic (Alemu et al., 2003) and Ugaritic (Zemánek, 2007) may also benefit from our annotation methodology.

After giving some background on the MHT in section 1, this paper describes the significance of morpho-syntactic dependencies in Modern Hebrew (section 2), and the dependency annotation scheme developed for the MHT (section 3). The rule mechanism that was used for automatically annotating dependencies in the MHT is described in section 4, and its manual evaluation is described in section 5.

1 Background - the Modern Hebrew Treebank

Semitic words are formed by concatenating a word stem and several clitics, including some prepositions, conjunctions, and the definiteness marker. These word parts are referred to as *word segments*. Word segments are assigned part-of-speech (POS) tags similar to POS tags of Indo-European words, with a fixed order of the possible POSs within the Semitic word. Word limits do not necessarily correspond to constituency structure: the segments of a single word may belong to several different constituents within the sentence. Thus, morphological analysis, and word segmentation in particular, is a precondition for syntactic analysis in Semitic lan-

guages (Sima'an et al., 2001; Diab et al., 2004). The Modern Hebrew Treebank (MHT) includes segmented and syntactically annotated text from *Ha'aretz* daily newspaper. The texts in the corpus cover several domains (news, society, politics, sports and business). Version 2 of the treebank (MHT2) is comprised of 6,501 sentences, 123,446 word tokens, and 162,829 word segments. Sections of text were automatically segmented into sentences and fed into an automatic morphological disambiguator for Hebrew (Segal, 2000), which outputs a suggested morphological analysis for each Hebrew word. This morphological annotation was automatically converted to word segments, with a POS tag assigned to each word-segment (Bar-Haim et al., 2007). This pre-processed text was then syntactically analyzed by human annotators, whose task was twofold: (i) correcting the automatically derived segmentation and POS-tagging; (ii) producing a syntactic analysis using the *Semtags* annotation-aiding tool (Bonnema, 1997). Following this stage, the last manually annotated version of the MHT was automatically enhanced by marking mother-daughter dependencies, which is in the focus of the present paper. The resulting treebank is publicly available[1] as MHT2.

A pilot syntactic annotation of 500 Hebrew sentences is described in Sima'an et al. (2001). Much of the annotation scheme in this pilot version of the MHT was maintained in MHT2, with some modifications. The MHT scheme applies flat annotations for structures where the Hebrew syntax allows a relaxed order of constituents. Notably, both VP complements and adjuncts are analyzed as VP external. Besides these departures from English grammatical conventions, many of the POS tags and annotation conventions were used in a similar way to the conventions of the Penn Treebank (Marcus et al., 1994).

2 The significance of mother-daughter dependencies in Hebrew syntactic analysis

The main new aspect of the annotation scheme of MHT2 is the analysis of "*mother-daughter*" dependencies: these dependencies are manifested through the (recursive) percolation of morphological information from one or more daughter nodes to their mother. We distinguish two different processes:

- *Dependency annotation* – marking a node according to the role it plays in determining its mother's features.
- *Feature percolation* – using these dependencies for feature percolation from daughter to mother.

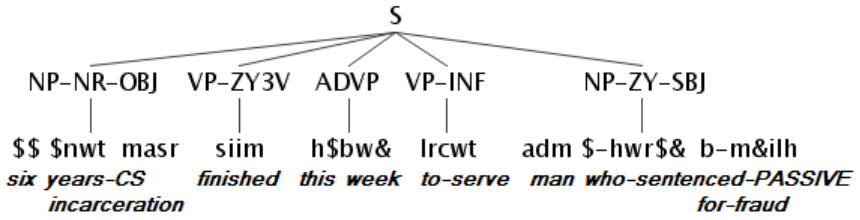
[1] `http://www.mila.cs.technion.ac.il/english/resources/corpora/treebank/ver2.0/index.html`

In English, the head constituent often determines the phrase's morphological features that must agree with the features of other constituents. Parsing systems often exploit such dependencies for disambiguation in head-driven methods (Charniak, 2001; Collins, 2003). In Semitic languages, due to some "free order" aspects of their syntax, mother-daughter dependencies are often syntactically critical. The Hebrew verb and its subject, internal arguments and adjuncts do not appear in a strict order in the sentence. As a result, agreement features percolated via mother-daughter dependencies often help to identify and disambiguate grammatical relations. For a recent demonstration that morpho-syntactic dependencies and agreement features significantly improve the performance of a non-lexicalized parser for Hebrew, see Tsarfaty and Sima'an (2007) and Tsarfaty and Sima'an (2008).

Mother-daughter dependencies in Semitic languages are often complicated by the *construct state* (CS) case configuration within complex nominals. The special properties of Hebrew CS nominals are well-documented in the linguistic literature (Glinert, 1989; Wintner, 2000; Danon, 2008). These constructions exhibit a syntactic relation between two adjacent nominals. The first part of the CS is the *construct noun*, which carries a special CS morphology. According to standard tests of selectional restrictions, the construct noun functions as the "semantic head" of the CS nominal. The second, obligatory, part in a CS configuration is the *post-construct NP*, which follows the CS noun. The head noun of the post-construct NP is morphologically unmarked. The number and gender features of CS nominals are determined by the construct noun. Importantly, however, syntactic definiteness is determined by the post-construct NP. Such dependencies on more than one daughter are common in MHT2, and in particular with construct states: 23.3% of all mother-daughter feature percolations in MHT2 are multiple daughter percolations, 30.8% of those multiple dependencies are in CS nominals.

As a simple example, consider the following Hebrew sentence with the morpho-syntactic annotation of its main constituents:[2]

(1)



"A man sentenced for fraud finished serving this week six years of incarceration"

[2]For our notation of agreement features and the Hebrew transliteration see tables A and B in the appendix.

Dependency tag	Used for marking:
DEP_HEAD	Percolation of all the agreement features
DEP_MAJOR	Percolation of all features except for definiteness or number features marked on sister nodes
DEP_DEFINITE	Percolation of the definiteness feature
DEP_NUMBER	Percolation of the number feature
DEP_ACCUSATIVE	Percolation of accusative case from the accusative marker 'AT'
DEP_HEAD_MULTIPLE	Percolation of agreement features from multiple sisters (for conjunctions)

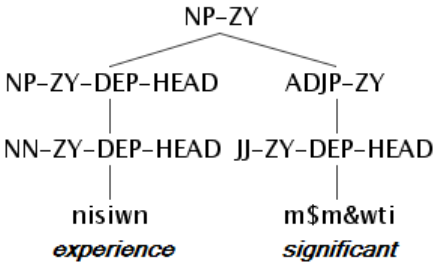
Table 1: Morpho-Syntactic Dependency Tags in MHT2

Sentence (1) contains two candidate noun phrases for being the subject of the matrix verb *siim* ("finished"). The main factor for classifying the second NP ("a man sentenced for fraud") as the subject of (1) is its singular masculine feature ('ZY'), which agrees with the singular masculine morphological form of the verb *siim* ("finished"). The 'ZY' feature of the subject, in turn, percolates from its head noun *adm* ("man"/"person"). In our work, this morpho-syntactic dependency relation between the subject NP node and the head noun is automatically recognized. Subsequently, the agreement features of the subject are automatically percolated from the manually annotated noun in the MHT.

3 The dependency annotation scheme

We mark mother-daughter dependencies using *dependency tags* that are added as additional features of syntactic categories. These tags encode the features that are percolated from a constituent to its mother node. Our scheme includes six types of dependency tags, listed and described in table 1.

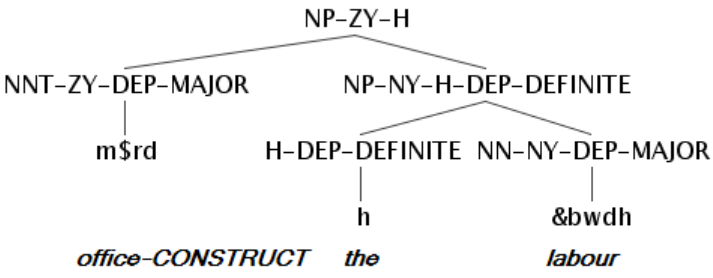
DEP_HEAD The tag DEP_HEAD is used for annotating the daughter node from which all features percolate to the mother node. In example (2) below, the words "experience" (*nisiwn*) and "significant" (*m$m&wti*) are heads of their respective NP and ADJP: they pass on their features, Z (masculine) and Y (singular), to the dominating phrasal node. In turn, the lower-level NP is tagged DEP_HEAD, since all of its morphological features are passed on to the main NP.



(2)

"A significant experience"

DEP_MAJOR and DEP_DEFINITE The tag DEP_MAJOR is used for annotating the daughter node from which most but not all features percolate to the mother node. In such structures, some of the features are percolated from nodes other than the head. This is the case in the aforementioned construct state nominals. We use the DEP_MAJOR tag to denote the percolation of all features but definiteness from the construct noun. The definiteness feature is inherited from the post-construct NP, which is annotated with the DEP_DEFINITE tag.



(3)

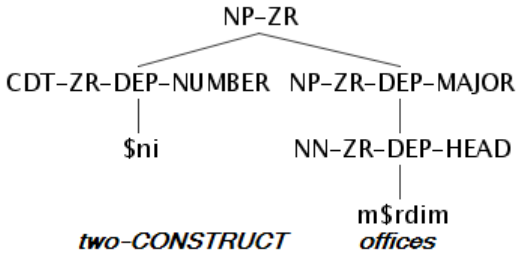
"The Labour Office"

Example (3) illustrates another use of the DEP_MAJOR dependency tag in our scheme, which is a result of the segmentation conventions of the MHT. As explained in Sima'an et al. (2001), the syntactic annotation scheme of the treebank analyzes definite articles as separate segments rather than as one of the nominal features of the head noun. Consequently, the definiteness feature of the definite noun *h-&bwdh* in (3) is inherited from the definiteness (*h*) segment. The other agreement features of this constituent are inherited from the noun *&bwdh*. The definiteness marker *h* is therefore annotated with the DEP_DEFINITE tag, and the noun *&bwdh* with the DEP_MAJOR tag.

DEP_NUMBER The dependency tag DEP_NUMBER is used for annotating a daughter node from which only the number feature percolates to the mother node.

Consider the following example, which illustrates the annotation of feature dependency within a numeral CS.

(4)

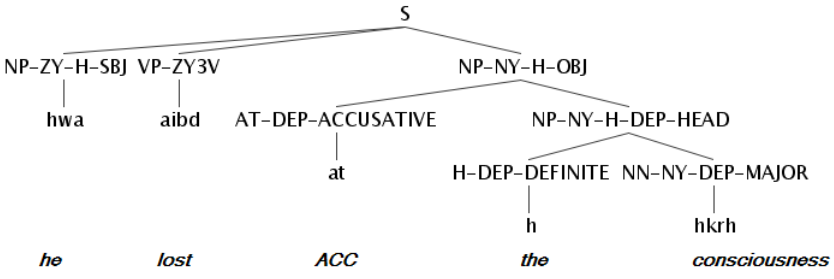


"Two offices"

The numeral item *$ni* ("two") has the marked construct morphology. However, unlike the nominal construct state in (3), this construct provides only the number feature (indicated by the DEP_NUMBER dependency tag); the head element *m$rdim* ("offices") determines the other features of the noun phrase (gender and definiteness), and is therefore tagged DEP_MAJOR.

DEP_ACCUSATIVE The tag DEP_ACCUSATIVE denotes the accusative marker 'AT', which requires the mother NP to be an object (have the OBJ feature). The following example illustrates this dependency tag.

(5)

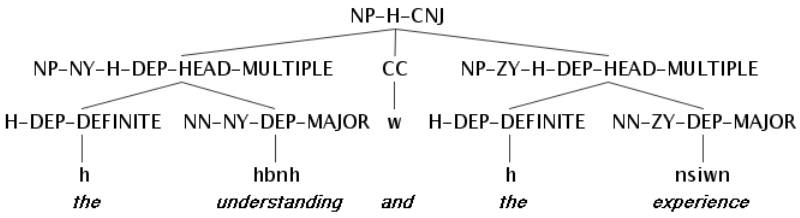


"He lost consciousness"

DEP_HEAD_MULTIPLE The dependency tag DEP_HEAD_MULTIPLE marks feature percolation from multiple coordinated constituents to the mother node. The features percolated depend upon the nature of the phrasal constituent. In an adjective phrase conjunction, the features of the conjoined adjectives have to match each other, and therefore all features are percolated from daughter adjectives to the mother conjunction with no conflict. In a nominal conjunction, however, the gender and number may differ between the coordinated noun phrases. Therefore only the definiteness feature is percolated in nominal conjunctions, as in the following

example:

(6)



"The understanding and the experience"

This example illustrates a limitation of the current annotation scheme: in such conjunctions, the scheme could have kept track of the fact that the dominating conjunction node inherits its plural and masculine feature values from both conjuncts. Such complex dependencies are not analyzed in MHT2.

4 Implementing the scheme: automatic dependency annotation and feature percolation

The annotation scheme presented in the previous section was manually coded as rules in XML format. These rules describe marking of mother-daughter dependencies in noun phrases, verb phrases, adjective phrases and prepositional phrases in the MHT. Further, the XML rules were processed by Python scripts that used them for adding morpho-syntactic dependencies and feature percolation to the MHT. The output of these scripts is the current MHT2. See Appendix C for some figures on the MHT2 treebank.

4.1 The annotation procedure

A linguist annotator divided the phrasal categories of the corpus into a small number of general cases. Each case defines the appropriate dependency scheme from those in Table 1, and the feature percolation that it sanctions. Overall, 24 dependency rules were written in XML format. Most of the rules were designed for NPs, where feature percolation depends on heterogeneous factors: especially the presence of a definite article, pronouns or proper names, coordination, embedding and construct states. After a pilot version of these rules was run on the corpus, the output was manually checked. Most of the rules were corrected accordingly, and subsequently run again on a second iteration to produce the current version of MHT2.

In the original annotation scheme, some special structural cases were defined as exceptions to the dependency rules. 1,942 such cases were automatically extracted

from the corpus and examined by a linguist annotator. However, in effect it turned out that only a few of these cases were incorrectly annotated by the automatic procedure. These errors were manually corrected. The estimated time for these manual corrections was 30 hours.

4.2 XML rules

The rules were coded in XML format and processed by Python scripts. The scripts applied the rules in a bottom-up order, so that to allow the percolation of features from daughter to mother nodes within structures. In total, 24 rules were developed. Consider for example the following informal rule describing dependency annotation and feature percolation for construct state NPs:

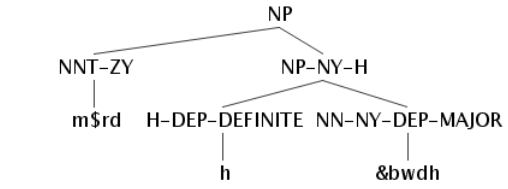
> Upon matching an NP which is comprised of a construct state noun (category 'NNT') followed by an embedded NP (the "post-construct" NP):
>
> - Copy all the properties of the construct noun except definiteness to the mother NP node
> - Set the dependency tag of the construct noun to 'DEP_MAJOR'
> - Copy the definiteness of the post-construct NP node to the mother NP node
> - Set the dependency of the post-construct NP node to 'DEP_DEFINITE'

The XML code corresponding to this rule is given below:

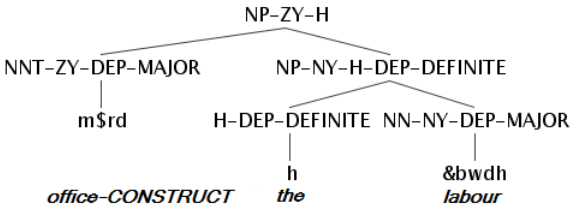
```
<rule name='5.1' dest='father'>
<struct>
<father label='NP'>
<child1 label='NNT' position='1' dep='DEP_MAJOR'>
<copy>-def</copy>
</child1>
<child2 label='NP' position='1' dep='DEP_DEFINITE'>
<copy>def</copy>
</child2>
</father>
</struct>
</rule>
```

By applying the Python scripts to this XML description of the rule and to structure (7) below, we generate analysis (8) for example (3) above.

(7)



(8)



"The Labour Office"

5 Correctness evaluation of dependency annotations

Evaluation of our dependency annotation was performed by selecting at random 50 sentences from the MHT2 (about 0.77% of the corpus' sentences). A linguist who was not involved in the rule-writing process checked these sentences for morpho-syntactic annotation, dependency annotation, feature percolation and syntactic agreement between constituents.

The 50 sentences contained on average 19.2 words and 25.3 word segments per sentence. Of the 1,227 dependency tags annotation in the analyses of these sentences, 837 dependency tags were on word segments and 390 on complex phrases. The errors that were detected were either in manual annotations or in the automatic application of the dependency scheme.

1. Incorrect manual POS or syntactic tag annotations – 3 errors.
2. Incorrect automatic dependency tag annotation – 6 errors.
3. Underspecification of features – 6 cases.

In addition, one incorrect percolation of features was detected in one of the "exceptional" cases mentioned above, where percolation was manually revised. Also the 6 incorrect dependency tag annotations were caused by mistakes in the manual syntactic annotation of a previous version of the MHT. Cases of feature underspecification were mainly due to the partial coverage of the rules on nominal

conjunctions. Another cause of feature underspecification was the percolation in certain structures of the bi-gender feature ("B") to the upper mother node.

From this error analysis we conclude that the automatic part of the annotation is highly reliable, and errors are only likely to happen when mistakes in manual annotation are detected.

6 Conclusion

While manual annotation of treebanks is known to be time consuming and error prone, a syntactically annotated corpus can be augmented with additional data based on linguistically informed schemes. In this work we examined one such case – dependency annotations and feature percolation in a Modern Hebrew treebank. The complexities of Semitic morpho-syntax, and especially the construct state, make this task considerably challenging. However, our conclusion from this work is that once syntactic constituency and categories are reliably annotated, augmenting them with dependency tags can be done using a robust automatic procedure, informed by solid linguistic analysis of the relevant constructions. We therefore believe that our contribution is not only useful for syntactically informed tasks in Modern Hebrew, but that it could also be of high value to on-going work on other Semitic resources, and notably syntactic resources for Arabic like the Penn Arabic Treebank.

A Agreement features

Gender	**Z=masculine, N=feminine, B=both**
Number	Y=singular, R=plural, B=both
Person	1,2,3
Tense	V=past, H=present, T=future, C=imperative
Definiteness	H=definite, U=underspecified

B Hebrew transliteration table

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
				&	$				@								
א	ב	צ	ד	ע	ש	ג	ה	י	ט	כ	ל	מ	נ	%	פ	ק	ר

S	T	U	V	W	X	Y	Z
ס	ת	″	ו	ח	ז		

C MHT2 figures

	Words	Segments	Segment dependencies	**Structural dependencies**
For total number of sentences	123,446	162,829	107,251	48,955
Average per sentence	19.0	25.0	16.5	7.5

Table 2: figures about the MHT2 (6,501 sentences)

Percolation levels	Number of features percolated	Percentage
1	124,844	72.035%
2	38,872	22.429%
3	8,116	4.683%
4	1,268	0.731%
5	198	0.114%
6	12	0.007%
Total	73,310	

Table 3: Distribution of the number of percolation levels

Dependency tag	Total number	Average per sentence
DEP_HEAD	103,062	16.0
DEP_MAJOR	25,600	4.0

Table 4: Number of MAJOR vs HEAD dependencies

Percolation from		Number of rule applications	Percentage (out of total number of rule applications)	Percentage total number of rule applications involving feature percolation)
No percolation		2,776	2%	-
Single child		93,192	69%	76.7%
Multiple children	Total:	28,413	21.1%	23.3%
	Construct states:	8,775	6.5%	7,2%
	Conjunctions:	1,761	1.3%	1,5%

Table 5: Distribution of rule applications

References

A. Alemu, L. Asker and G. Eriksson. 2003. An Empirical Approach to Building an Amharic Treebank. *TLT 2003*. `http://w3.msi.vxu.se/~rics/TLT2003/doc/alemu_et_al.pdf`

R. Bar-Haim, K. Sima'an, and Y. Winter. 2007. Part-of-Speech Tagging of Modern Hebrew Text. In *Journal of Natural Language Engineering.*

R. Bonnema. 1997. Data Oriented Semantics. Master's thesis.

M. Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4).

E. Charniak. 2001. Immediate-Head Parsing for Language Models. In *Proceedings of ACL 2001*: 116-123.

G. Danon. 2008. Definiteness spreading in the Hebrew construct state. Lingua 118(7), 872-906.

M Diab, K. Hacioglu, and D. Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In Marcu, Dumais and Salim Roukos, editors, *HLT-NAACL 2004*: Short Papers, pages 149-152.

L. Glinert. 1989. The Grammar of Modern Hebrew. *Cambridge University Press*.

M. Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR International Conference on Arabic Language Resources and Tools.*

S. Mansour, K. Sima'an and Y. Winter. 2007. Smoothing a Lexicon-based POS Tagger for Arabic and Hebrew. *ACL2007 Workshop on Computational Approaches to Semitic Languages*.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate-Argument Structure.

E. Segal. 2000. Hebrew Morphological Analyzer for Hebrew undotted texts. Master's thesis.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a Tree-Bank of Modern Hebrew Text. In *Traitment Automatique des Langues*.

O. Smrž, V. Bielický, I. Kouřilová, J. Kráčmar, J. Hajič, P. Zemánek. 2008. Prague Arabic Dependency Treebank: A Word on the Million Words. In *Proceedings of the LREC-2008 workshop on HLT & NLP within the Arabic world: Arabic Language and local languages processing*.

R. Tsarfaty, and K. Sima'an. 2007. Accurate Unlexicalized Parsing for Modern Hebrew. In *Proceedings of Text, Speech and Dialog* (TSD).

R. Tsarfaty and K. Sima'an. 2007. Three-Dimensional Parametrization for Parsing Morphologically Rich Languages. In *Proceedings of the International Conference on Parsing Technologies* (IWPT).

R. Tsarfaty and K. Sima'an. 2008. Relational-Realizational Parsing. In *Proceedings of The 22nd International Conference on Computational Linguistics (CoLing).*

S. Wintner. 2000 Definiteness in the Hebrew Noun Phrase. In *Journal of Linguistics*, 36:319-363.

P. Zemánek. 2007. A Treebank of Ugaritic. Annotating Fragmentary Attested Languages. In *TLT 2007*. `http://tlt07.uib.no/papers/9.pdf`

The PASSAGE Syntactic Representation

P. Paroubek[+] E. de la Clergerie[*] S. Loiseau[+] A. Vilnat[+] G. Francopoulo[$]
[+]LIMSI-CNRS [*]ALPAGE-INRIA-U. Paris 7 [$]TAGMATICA
E-mail: [+]{pap,sloiseau,anne.vilnat}@limsi.fr
[*]Eric.De_La_Clergerie@inria.fr
[$]gil.francopoulo@tagmatica.com

Abstract

We present the PASSAGE syntactic representation based on syntactic relations, initially developed for French in the scope of national evaluation campaigns. After a brief presentation of the non-nested chunks and syntactic relations of PASSAGE, we reuse the comparison elements that Marneffe and Manning have selected to compare the Standford typed dependencies (SD) against the GR and PARC representations, and show that PASSAGE is for a large part compatible compatible with these representation, standing closer to GR than to SD. After a presentation of the collaborative software support for PASSAGE representation, we conclude on some essential characteristics that pivot representation for syntax should exhibit.

1 Introduction

The work presented in the paper takes place in the context of PASSAGE [1][10][5] , a 3-year French action with the following main tasks:

- automatically annotating a French corpus of about 100 million words using 10 parsers;
- merging the resulting annotations using a combination algorithm in order to improve annotation quality ;
- manually building a reference annotated subcorpus (around 400,000 words),
- performing knowledge acquisition experiments from combined annotations,

[1](ANR-06-MDCA-013)(*Produire des annotations syntaxiques à grande échelle* – Large Scale Production of Syntactic Annotations), (2007–2009)

- running two parsing evaluation campaigns on the model of the EASy French evaluation campaign [7]. The first campaign was run during October 2007, with 10 parsers. From the data collected on this occasion, we extracted parameters for the the combination algorithm. The second campaign, at the end of PASSAGE (2009), should provide information about the evolutions of the parsers during the project.

The representation used in PASSAGE [2] is based on the EASy representation whose first version was crafted in an experimental project PEAS [4], with inspiration taken from the propositions of [2]. The representation has been completed with the input of all the actors involved in the EASy evaluation campaign (both parsers' developers and corpus providers) and refined with the input of PASSAGE participants. This formalism aims at making it possible to compare all kinds of syntactic annotation (shallow or deep parsing, complete or partial analysis), without giving any advantage to any particular approach. It has six kinds of non-nested chunks [3] and 14 kinds of relations Some of them are illustrated in Figure 1. Like [1], the annotation formalism allows the annotation of minimal, continuous and non-nested chunks, as well as the encoding of relations wich represent syntactic functions. These relations (all of them being binary, except for the ternary coordination) have sources and targets which may be either word forms or chunks.and either extra-chunks or intra-chunk. The direction between source and target has been arbitrarily defined according to custom, this constitutes a minor point since the essential information lies in the label of the relation because we do not require the annotations to build trees but content with graphs. Note that the PASSAGE annotation formalism does not postulate any explicit *head*, see section 4.

2 Chunk annotation

For the PASSAGE campaigns, 6 kinds of chunks have been considered as illustrated below and in the table 2. These chunks are minimal and not embedded. The reason of this choice is to allow the evaluation of different kinds of parsers, as explained previously.

[2] http://www.limsi.fr/Recherche/CORVAL/PASSAGE/eval_1/2007_10_05 PEAS_reference_annotations_v11.12.html

[3] Defined in the Data Category Registry of ISO 12620 as a flat sequence of words typically containing more than one word, see http://syntax.inist.fr.

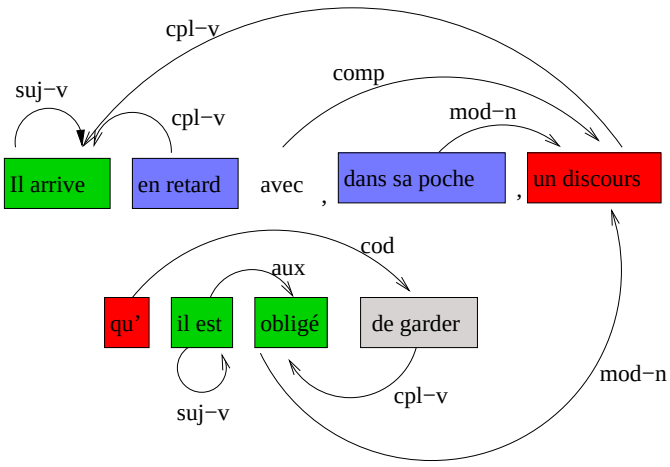


Figure 1: Example of a sentence annotated in chunks and relations. *He arrives late with, in his pocket, a discourse he is prevented to pronounce*

- the noun phrase (**GN** for *Groupe Nominal*): a noun preceded by a determiner and/or by an adjective with its own modifiers, a proper noun or a pronoun;
- the prepositional phrase (**GP**, for *groupe prépositionnel*): a preposition and the GN it introduces, a contracted determiner and preposition, followed by the introduced GN, a preposition followed by an adverb or a relative pronoun replacing a GP; in some constructions, the preposition is separated from the GN (as in the example of Figure 1), the corresponding annotation will be explained below;
- the verb kernel (**NV** for *noyau verbal*) includes a verb, the clitic pronouns[4] (always closed to the verb) and possible particles attached to it. Verb kernels may have different forms: conjugated tense, present or past participle, or infinitive. When the conjugation produces compound forms, distinct NVs are identified for each part of the compound;
- the adjective phrase (**GA** for *groupe adjectival*) contains an adjective when it is not placed before the noun, or past or present participles when they are used as adjectives;
- the adverb phrase (**GR** for *groupe adverbial*) contains an adverb;
- the verb phrase introduced by a preposition (**PV** *noyau verbal à préposition*): a verb kernel with a non-inflected verb introduced by a preposition. Some modifiers or adverbs may also be included in a PV.

[4]The French personal pronouns except disjunctive ones are all clitics.

GN	- [la très grande porte] (*the very big door*), [Rouletabille] - [eux] (*they*), [qui] (*who*)
GP	- [de la chambre] (*from the bedroom*), [du pavillon] (*from the lodge*) - [de là] (*from there*), [dont] (*whose*)
NV	- [j'entends] (*I hear*), [on ne l'entend] plus (*we hear her no more*) - [désobéissant] à leurs parents (*disobeying their parents*) - Il [ne veut] pas [venir] (*He doesn't want to come*) - [ils etaient] [fermés] (*they were closed*)
GA	- les barreaux [intacts] (*the intact bars*) - la solution [retenue] fut... (*the chosen solution was...*) - les enfants [désobéissants] (*the disobeying children*)
GR	- [aussi] (*also*), vous n'auriez [pas] (*you would not*)
PV	- [pour aller] à Paris (*to go to Paris*), [de vraiment bouger] (*to really move*)

Table 1: [Chunk examples], (with their English translation).

3 Syntactic relation annotation

The dependencies establish all the links between the chunks described above. All participants, corpus providers and campaign organizers agreed on a list of 14 kinds of dependencies listed below:

- subject-verb (**SUJ-V**): may be inside the same NV as between *elle* and *était* in *elle était* (*she was*), or between a GN and a NV: *Mademoiselle appelait* (*Miss was calling*);
- auxiliary-verb (**AUX-V**), between two NVs: *on a construit une maison* (*we have built a house*);
- attribute-subject/object (**ATB-SO**): between the attribute and the verb kernel, and precising that the attribute is relative to (a) the subject: *il est grand* (*he is tall*), or (b) the object: *il trouve cette explication étrange* (he finds this explanation strange);
- 3 kinds of dependencies between the verb and complements or modifiers
  - direct object-verb (**COD-V**): *on a construit la première automobile* (*we have built the first car*);
  - complement-verb (**CPL-V**): in case of adjuncts or indirect objects: *en quelle année a-t on construit* la première automobile (*In which year did we build the first car*);

  - modifier-verb (**MOD-V**):for not mandatory modifiers, as adverbs or adjunct clauses:
    Jean dort quand la nuit tombe (*Jean sleeps when the night falls*);

- complementor (**COMP**): to link the introducer and the verb kernel of a subordinate clause: *Je pense qu' il viendra* (*I think that he will come*); it is also used to link a preposition and a noun phrase when they are not contiguous, preventing us from annotating them as a GP as in:
  avec dans sa poche un discours (see Figure1);
- different modifiers to relate to the noun (resp. adjective, adverb or preposition) all the chunks which modify it:
  - modifier-noun (**MOD-N**): for the adjective, the genitive, the relative clause: *l'unique fenêtre* (*the unique window*); *la porte de la chambre* (*the bedroom door*);
  - modifier-adjective (**MOD-A**): *la très belle collection* (*the very impressive collection*) or *elle est fière de son fils* (*she is proud of her son*);
  - modifier-adverb (**MOD-R**): *elle vient très gentiment* (*she comes very kindly*);
  - modifier-preposition (**MOD-P**): *elle vient peu avant lui* (*she comes little before him*);
- coordination (**COORD**): to relate the coordination and the coordinated elements, as between *Pierre*, *Paul* and *et* in *Pierre et Paul arrivent* (*Pierre and Paul are arriving*);
- apposition (**APP**): to link the elements which are placed side by side, when they refer to the same object: *Le député Yves Tavernier...* (*the MP Yves Tavernier...*);
- juxtaposition (**JUXT**): to link chunks which are neither coordinated nor in an apposition relation, as in an enumeration. It also links clauses as in:
  on ne l' entendait plus ... elle était peut-être morte (*we did not hear her any more... perhaps she was dead*).

Some examples are provided in Figure 1 or in section 4.

4 Some elements of comparison with SD, GR and PARC

In this section, we consider how the PASSAGE annotation scheme addresses the comparison elements that [3] used to ascertain the position of the Standford typed

dependencies (SD) against the GR and PARC representations. Since SD was designed with task based evaluation as opposed to intrinsic evaluation, we found these elements of comparison to be more likely to enhance the contrasts between the two representations. We briefly address: argument/adjunct distinction, NP-internal relations, noun-modifier dependencies, head identification, the SD dependency collapsing mechanism, preposition modifiers, arity of the syntactic relations and the choice of having a representation more oriented towards syntax or semantics.

The SD scheme is not concerned with the argument/adjunct distinction but in contrast it includes many NP-internal relations such as *appos* (appositive modifier), *nn* (noun compound), *num* (numeric modifier), number (element of compound number) and *abbrev* (abbreviation). The following example, taken from [3] *"I feel like a litle kid", says a gleeful Alex de Castro, a car salesman, who has stopped by a workout of the Suns to slip six Campaneris cards to the Great Man Himself to be autographed. (WSJ-R)* yields the following relations which we have completed with the PASSAGE ones in table 4. It shows that PASSAGE includes dependencies similar to the other schemes, but of a coarser grain for what concerns the dependencies source/target text extents. PASSAGE was designed with the aim of addressing only the essential level of syntactic functions, leaving aside finer grain relations like the determiner one and information more related to lexical issues like those addressed by SD with *element of compound number* or *abbreviation* or by PARC with verb tense and aspect, noun number and person and named entities types. PASSAGE was not designed either to address semantics since the conditions of its creation representation were intrinsic parser evaluation.

Note that with PASSAGE, intra-chunk relations such as the MOD-N relation between *gleeful* and *Alex* can only address single word forms and not chunks as is the general case. This is because PASSAGE does not allow the nesting of chunks. In the case the of MOD-N relation, we preferred in PASSAGE to have a nominal consituent holding the apposed adjective and an intra-chunk MOD-N relation instead of an adjectival and nominal chunk linked by a MOD-N relation because adjectives occuring before the noun are much less frequent in French than those occuring after and the corresponding syntactic structure is generally straightforward.

The last remarkable point about Table 4 lies in the label variability among the representation schemes for the *to slip - stopped* dependency. PASSAGE does not have the notion of head, instead it uses its six basic chunks presented in section 1 to restrict the portion of text where a head can possibly occur. Since the notion of head is controversial, see for instance [9], we did not want to have in the PASSAGE formalism any explicit reference to this notion, not even in the documentation describing how to annotate chunks. Our initial design choice was motivated by the wish to have a syntactic representation as simple as possible to ease up the annotation task and to be able to compare parsing schemes which have heads against

ones which do not. Note that PASSAGE does not forbid to address single words as target dependency, so a representation scheme with heads can be mapped directly onto PASSAGE, at the price that if a comparison is done with another annotation which has chunks, the precise location of the head will be lost in the process, since it will be assimilated with the scope of the enclosing chunk. So instead of identifying *effective* as head of the quoted phrase like SD in the example used by [3] *Considered as a whole, Mr Lane said, the filings required under the proposed rules "will be at leas as* **effective***, if not more so, for investors following transactions"*

Scheme	Relation
SD	appos(Castro, salesman)
PARC	appos(Alex de Castro, salesman)
GR	adjunct(Castro, salesman)
PASSAGE	appos(FIRST:[a gleeful Alex de Castro]^{GN} APPOSED:[a car salesman]^{GN})
SD, PARC, GR	num(card, six)
SD	nn(cards, Campaneris)
PASSAGE	mod-n(MODIFIER:six, NOUN:cards)
SD	amod(Castro, gleeful)
PARC	adjunct(Alex de Castro, gleeful)
PASSAGE	mod-n(MODIFIER:gleeful, NOUN:Alex)
SD	amod(kid, little)
PARC	adjunct(kid, little)
PASSAGE	mod-n(MODIFIER:little, NOUN:kid)
SD	xcomp(stop, slip)
PARC	adjunct(stop, slip)
PASSAGE	mod-v(MODIFIER:[to slip]^{NV}, VERB:[stopped]^{NV})
SD	prep_of(workout, Suns)
PARC	adjunct(workout, of)
PASSAGE	MOD-N(MODIFIER:[of the Suns]^{GP}, NOUN:[by a workout]^{GP})

Table 2: Comparing SD, PARC, GR and PASSAGE, an example.

(WSJ-R) or identifying *be* as head like GR, PASSAGE will identify *will* as a target of a subject-verb relation originating at *filings*. Here we see that PASSAGE is nearer GR, we could say in a way more oriented toward the coding of explicit syntax, while SD tends to address more semantic aspects. Lastly, it is not because in some cases PASSAGE chunks identify to heads when they have a size 1 (e.g. with some verbal chunks) that one could systematize such correspondence, for instance identifying a head in a verbal chunk become problematic when it contains also the clitic pronoun. In the same idea of annotating an explicit link between content words, SD proposes a collapsing mechanism for dependencies involving prepositions, in the previous example, instead of having two relations, as will be the case with GR and PARC, SD may have only one given in the table 4. Here PASSAGE

Scheme	Relation
PARC	adjunct(workout, of), obj(of, Suns)
GR	ncmod(workout, of), dobj(of, Suns)
SD	prep_of(workout, Suns)
PASSAGE	MOD-N(MODIFIER:[of the Suns]^{GP},NOUN:[by a workout]^{GP})

Table 3: Collapsing dependencies with SD

is closer to the SD representation, since it will also have a single MOD-N relation between the two chunks identified as prepositional chunks (GP tags above).

The fact that SD leans toward semantics and PASSAGE does toward syntax can also be seen in the example [3]: *A similar technique is almost impossible to apply to other crops, such as cotton, soybean and rice (WSJ-R)* for which SD gives direct *prep_such_as* links between *crops* and all the coordination elements while this information can only be accessed through indirect links with PASSAGE as shown in figure 2.

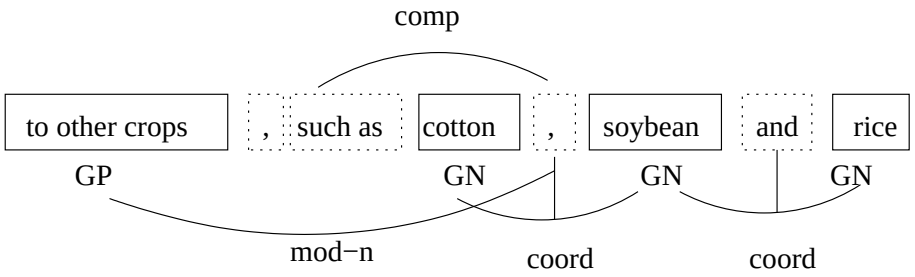


Figure 2: PASSAGE annotation of the “crops” example

One point where PASSAGE is more faithful to linguistic interpretation that SD concerns preposition modifiers; while PASSAGE has a *MOD-P* relation for the purpose (see 4), SD binds the preposition modifier to the head of the clause in which they appear and not on the preposition itself [3].

Although PASSAGE is very close to GR, it does not annotate explicitly passive constructions (while GR, PARC and SD do, for a comparison see [8]) or deep subjects: a deliberate choice to stick to explicit first level interpretation for the initial trial that was the first PASSAGE evaluation campaign. But extra annotations are already in discussion for the version that will be used for semantic extraction at the end of the PASSAGE project.

[3] argue that to have only binary relations (all dependencies are a triples, a grammatical relation, the head and the dependent), makes the representation more readable, easier to encode in software and to map to semantic WEB representation such as OWL and RDF triples. If we agree with the later points, we chose in PASSAGE to have a surface form for the annotations which have some ternary relations because they are closer to the intuitive syntactic representation (in general a coordination involves three elements, a conjunction and two conjuncts). Nevertheless, all these ternary relations, except the coordination, have a third argument which is used only for specifying a subtype of the relation, like for instance the distinction between subject-attribute versus object-attribute for the verb-attribute relation and could be automatically transformed into a canonical binary representation by defining new relations for the subtypes. PASSAGE coordination relation could also be transformed automatically into two coordination relations, one linking the left part and the conjunction and the other for the right part. Here we address "syntactic sugar" issues, since we allow instances of the coordination relation where the left part is empty, for instance when a coordination conjunction is used to start a sentence, a phenomena often encountered in speech transcriptions.

Another characteristic of PASSAGE that we see as an advantage over SD for what concerns the annotation task, comes from having a representation as intuitive as possible in terms of syntax; we are not faced with the dilemma of either altering the semantics of a sentence or duplicating verbs when dealing with preposition conjunction and preposition collapsing. For the sentence *Bill went over the river and right through the woods* from [3], SD will duplicate the verb *went* while PASSAGE will straightforwardly represent the initial syntactic information preserving uniformity of handling of the two GPs at the price of an indirection throughout the coordination relation to represent the link between *went* and its two adjuncts, see table 4. We see here that PASSAGE adopts the "Prague style" for annotating coordination, see [6] for a more detailed discussion. 4.

Scheme	Relation
SD	a prep_over(wen-2, river-5) prep_through(went-2', woods-10) conj_and(went-2, went-2')
PASSAGE	MOD-V(MODIFIER: and, VERB:[went]^{NV}) COORD([over the river]^{GP}, [through the woods]^{GP}) MOD-P(MODIFIER:right, PREPOSITION:through)

Table 4: Preposition collapsing with SD

5 EASYREF the collaborative software support for PASSAGE

EASYREF is a collaborative WEB browsing/editing/versioning software developed by INRIA for corpus annotated with the PASSAGE representation. The display uses a linear representation of the sentences with color-coded chunks above the forms. The idea was extended to dependencies, represented on several lines below the forms using color codes related to their type and span given by their anchors. One may select which kinds of dependencies are to be displayed and when moving the mouse over a dependency, its anchors are highlighted and a tooltip box is displayed providing more detailed information. For a given sentence, it is possible to show or hide additional pieces of information, such as the list of its associated bug reports, the history of its revisions and a list of potential annotation errors automatically detected by EASYREF.

Sentences may be searched using various administrative and linguistic criteria; e.g. one may search for all the sentences with potential errors but no bug reports, or for sentences with reports but no corrections. A more linguistic query such as "\verb+évaluer@NV les@GN+" would return all the sentences where the word "évaluer" (*evaluate*) in a NV chunk is followed by "les" (*the*) in a GN chunk. Linguistic queries are applied as regular expressions on a linear representation of both text and chunk annotations. Querying with syntactic relations is under development along with an extension of the PASSAGE annotation scheme which will have lemma, morphosyntactic tags, a fine grained representation of token/word form segmentation and nested chunks.

It is also possible to compare two sets of annotations coming from two different parsers, for instance, as illustrated in Figure 3. Color codes provide an easy identification of mismatching chunks. Comparing dependencies is more complex:

both sets of dependencies are actually mixed in the display, here the text color and weight indicate the status of the dependencies: shared or belonging only to one set.

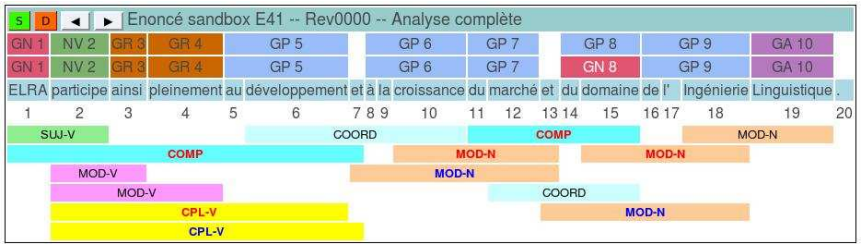


Figure 3: Comparing two annotation sets.

6 Conclusion

We have presented the PASSAGE syntactic representation based on syntactic relations, initially developed for French in the scope of national evaluation campaigns and shown that it stands closer to GR than SD or PARC. Software support for the representation is provided by EASYREF, a collaborative WEB browser/editor that we have presented. PASSAGE representation contributes to the ongoing debate on a common pivot formalism for syntactic information by proposing to replace the requirement of precise head localization with one level of chunking in complement of its functional dependencies.

7 Acknowledgements

We wish to thank the organizers and participants to the workshop on Cross-Framework and Cross-Domain Parser Evaluation at COLING 2008 for their warm welcome and useful suggestions.

References

[1] S. Buchholz, J. Veenstra, and W. Daelemans. Cascaded grammatical relation assignment. In *In proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 239–246, 1999.

[2] J. Carroll, D. Lin, D. Prescher, and H. Uszkoreit. Proceedings of the workshop "Beyond Parseval - Toward improved evaluation measures for parsing systems". In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Spain, 2002.

[3] M.-C. de Marneffe and C. D. Manning. The Stanford typed dependencies representation. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation at COLING 2008*, pages 1–8, Manchester, August 2008. Association for Computational Linguistics.

[4] V. Gendner, G. Illouz, M. Jardino, L. Monceaux, P. Paroubek, I. Robba, and A. Vilnat. PEAS the first instanciation of a comparative framework for evaluating parsers of French. In *Proceedings of the 10^{th} Conference of the European Chapter fo the Association for Computational Linguistics*, pages 95–98, Budapest, Hungary, April 2003. ACL. Companion Volume.

[5] E. de la Clergerie, O. Hamon, D. Mostefa, C. Ayache, P. Paroubek, and A. Vilnat. Passage: from French parser evaluation to large sized treebank. In ELRA, editor, *In proceedings of the sixth international conference on Language Resources and Evaluation (LREC)*, Marrakech, Morroco, May 2008.

[6] J. Nilsson, J. Nivre, and J. Hall. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia, July 2006. Association for Computational Linguistics.

[7] A. Vilnat C. Ayache P. Paroubek, I. Robba. Data, annotations and measures in EASY - the evaluation campaign for parsers of French. In ELRA, editor, *In proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, pages 315–320, Genoa, Italy, May 2006.

[8] L. Rimell and S. Clark. Constructing a parser evaluation scheme. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation at COLING 2008*, pages 44–50, Manchester, August 2008. Association for Computational Linguistics.

[9] Yoav Seginer. *Learning Syntactic Structure*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, October 2007. ILLC Dissertation Series DS-2007-05.

[10] A. Vilnat, G. Francopoulo, O. Hamon, S. Loiseau, P. Paroubek, and E. Villemonte de la Clergerie. Large scale production of syntactic annotation to move forward. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation (PE 2008) in conjuction with COLING*, pages pp 36–43, Manchester, August 2008.

The Distribution of Weak and Strong Object Reflexives in Dutch

Gosse Bouma and Jennifer Spenader

Information Science
University of Groningen
g.bouma@rug.nl

Artificial Intelligence
University of Groningen
j.k.spenader@rug.nl

Abstract

We use a syntactically annotated corpus to study the distribution of strong and weak reflexive objects in Dutch. Whereas previous work was limited to a small set of accidental reflexive verbs, we look at all transitive verbs in the corpus. We use subcategorization frames to approximate verb senses. We show that comparing the rate of pronominal usage to reflexive usage is a better predictor of strong or weak reflexive choice tendencies (giving a correlation of 33%) than considering all objects, confirming a suggestion by Haspelmath (2004). We also show that the automatic method gives results comparable to those for the semi-automatically collected data in Hendriks, Spenader, and Smits (2008).

1 Introduction

If a verb is used reflexively in Dutch, two forms of the reflexive pronoun are available. This is illustrated for the third person form in the examples below.

(1) a. Brouwers schaamt **zich/*zichzelf** voor zijn schrijverschap.
Brouwers is ashamed of his writing
b. Duitsland volgt **zichzelf** niet op als Europees kampioen.
Germany does not succeed itself as European champion
c. Wie **zich/zichzelf** niet juist introduceert, valt af.
Everyone who does not introduce himself properly, is out.

The choice between *zich* and *zichzelf* depends on the verb. Generally three groups of verbs are distinguished. Inherent reflexives are claimed to never occur with a non-reflexive argument, and as a reflexive argument are claimed to use *zich*

exclusively, (1a). Non-reflexive verbs seldom, if ever occur with a reflexive argument. If they do however, they can only take *zichzelf* as a reflexive argument (1b). Accidental reflexives can be used with both *zich* and *zichzelf*, (1c). Accidental reflexive verbs vary widely as to the frequency with which they occur with both arguments and it is this distribution that we would like to explain.

What exactly governs the choice between the weak and strong forms of a reflexive in the case of accidental reflexive verbs is largely unclear. The influential theory of Reinhart and Reuland (1993) explains the distribution as the surface realization of two different ways of reflexive coding. An accidental reflexive that can be realized with both *zich* and *zichzelf* is actually ambiguous between an inherent reflexive and an accidental reflexive (which always is realized with *zichzelf*). An alternative approach is that of Haspelmath (2004), Smits, Hendriks, and Spenader (2007), and Hendriks, Spenader, and Smits (2008), who have claimed that the distribution of weak vs. strong reflexive object pronouns correlates with the proportion of events described by the verb that are self-directed vs. other-directed.

In this paper we investigate to what extent a broad corpus investigation provides evidence for this claim. For each verb sense, we count how often it occurs with a strong or weak reflexive, or with another object. As many verbs occur rarely with a reflexive, a large amount of (parsed) data is required. We use a 470 M word Dutch corpus, syntactically analyzed using the Alpino-parser (van Noord, 2006) and use the results to make observations about reflexive use in general, the utility of large, parsed data sets, as well as the limits of a purely syntactic, unsupervised approach.

2 Previous Work

Haspelmath (2004), Smits, Hendriks, and Spenader (2007), and Hendriks, Spenader, and Smits (2008) have claimed that the distribution of weak vs. strong reflexive object pronouns (i.e. reflexives that are the object of a verb) correlates with the proportion of events described by the verb that are self-directed vs. other-directed. The claim is that if a verb is rarely used to express self-directed events, there will be a tendency to use the strong reflexive form when it is used reflexively to signal this marked use of the verb. The assumption behind the claim is that when the expectation that a given action will be self-directed is weak, emphasis on the reflexive argument is preferred, so the strong reflexive is used. Such emphasis is less likely if the verb is used with a self-directed meaning relatively often, and therefore the weak reflexive, which is shorter and should otherwise always be preferable, will be sufficient. This is in line with the claim that inherent reflexives

only occur with weak reflexives, since they only occur with reflexive meaning.[1]

Our research builds upon the work in Smits, Hendriks, and Spenader (2007) and Hendriks, Spenader, and Smits (2008), who studied the distribution of reflexive vs. nonreflexive use and the choice for a weak or strong form for 45 Dutch transitive verbs. Smits, Hendriks, and Spenader (2007) found a linear correlation between reflexive and non-reflexive usage (counting all third person NPs) for 21 % of the data in an 80 M word corpus (parsed using Alpino) for the verbs sufficiently frequent in the corpus. By combining this with judgement data, they were able to obtain an 83% correlation. Hendriks, Spenader, and Smits (2008), using a 300 M word corpus and 32 verbs obtained a correlation of 28% and a correlation of 30% when first and second person reflexives were included. Haspelmath (2004) suggests that only the ratio of pronominal objects to reflexive objects is relevant for determining the degree to which a verb is introverted (tends to describe self-directed events) or extroverted (tends to describe other-directed events). Hendriks, Spenader, and Smits (2008) found that the model proposed by Haspelmath yielded a correlation of 45%. However, they had no explanation as to why counting pronominal objects only gave more accurate results.

The research reported below differs from the approach of Hendriks, Spenader, and Smits (2008) in that we attempt to first empirically identify accidental reflexive verbs among all verbs in the corpus, and then use this very large set to test the different models of reflexive choice. The larger set of verbs may give us a more complete picture, but also forces us to adopt a fully automatic method for data collection, as we cannot afford to judge data individually for errors or unintended readings. In general, different senses of a verb may have very different tendencies for being used with self-directed activities. We therefore distinguish verbs by their different subcategorization frames in order to approximate verb senses.

3 Data Collection

We are interested in frequency estimates of the reflexive vs. nonreflexive use of the set of accidental reflexive verbs. Distinguishing accidental reflexives from inherent reflexives and non-reflexives is therefore crucial. A major problem is that most verbs are extremely ambiguous and simply checking if a verb can be used with a nonreflexive object or not is not sufficient:

[1] Note however that many inherent reflexives, like *zich herinneren*, (to remember) or *zich verspreiden*, (to spread out), can't really be characterized as being self-directed actions because the reflexive object doesn't seem to have a thematic role.

(2) a. De bedrijven maakten foute rekeningen op
The companies produced wrong bills
b. De schelpdieren maken al het voedsel op
The shellfish take all the food
c. Als ik 240 rijd, kan mijn assistente zich rustig opmaken
If I drive 240, my assistent can still put make-up on
d. De showbizz maakt zich op voor het huwelijk van het jaar
The showbizz prepares itself for the marriage of the year

The senses of *opmaken* illustrated in (2a) and in (2b) can hardly be used reflexively, the sense in (2c) can easily be used with a reflexive, while the sense in (2d) is inherently reflexive. Obviously, counting the frequency with which a verb occurs with an nonreflexive or reflexive object, without taking these differences in meaning into account, leads to noisy results. On the other hand, the parser does not annotate word senses, so we cannot automatically produce counts per verb sense.

The lexicalist nature of the Alpino-grammar implies that detailed verbal subcategorization frames are used to determine which complements a verb can combine with. By taking subcategorization frames into account some word sense distinctions can be identified. The inherent reflexive use of *opmaken* (2d), for instance, can be distinguished from the other senses by the fact that it subcategorizes for a PP-complement headed by the preposition *voor*.

Collecting counts for each pair of a verbal root + subcategorization frame is more precise than collecting counts per verbal root, but is still imperfect, as it fails to distinguish between verbal word senses with identical subcategorization frames. Verbs that have both an inherent reflexive use and an accidental reflexive use, for instance, are still problematic. (3a) illustrates a, highly frequent, idiomatic use of the verb *bedruipen*, which is inherently reflexive. Its meaning is clearly different from, although perhaps related to, the normal transitive use of *bedruipen* in (3b) (which is hardly found in the corpus).

(3) a. De vereningen kunnen zich met sponsoring bedruipen
The organisations can support themselves with sponsorships
b. Hij bedruipt een geitenkaasje met tijmhoning
He drips honey on a goat cheese

If *bedruipen* occurs with a reflexive, the parser has to choose between two verbal subcategorization frames: inherent reflexive or ordinary transitive. This choice is difficult, especially if the verb occurs with *zichzelf*. The inherent reflexive use is far more frequent than the ordinary transitive use. Nevertheless, in the case of *zichzelf*, the parser has a preference for using the ordinary transitive subcategoriza-

tion frame, instead of the frame associated with the inherent reflexive use.[2] This is unsurprising: strong reflexives in general do not occur with inherent reflexives. However, in ambiguous cases like this, this preference leads to inaccurate data. To avoid this problem, we discarded counts for all verb+subcategorization frames for which the parser has an alternative that differs from the current pair only w.r.t. the question whether the object obligatorily has to be a reflexive or not. This means that approximately 20% of the data is discarded.

Finally, we also decided to skip all occurrences of verbs that are used in passive sentences, or as complement of *laten*.

(4) a. De opstandelingen werden ontwapend
The rebels were disarmed
b. De kinderen laten zich niet dwingen
The children do not let themselves be forced

In passives, the object of the main verb appears as the subject of the passive auxiliary. In this position reflexives cannot be used. In sentences with *laten*, a reflexive may appear as the object of the embedded verb. This reflexive is interpreted as coreferential with the subject of *laten*, but it is unclear if it is also coreferential with the (unexpressed) subject of the embedded verb.

We used the 470 M word Twente News Corpus (TwNC), made up of the text of Dutch newspapers from the period 1994-2005 (Ordelman et al., 2007), which was parsed automatically with the Alpino-parser. Using the technology described in Bouma and Kloosterman (2007), we searched the corpus exhaustively for all occurrences of a verb with an object and a third person subject, and registered whether the object was *zich, zichzelf*, a (non-reflexive) pronoun, or a regular NP. We extracted 12 M verb-object tuples.

4 Distribution of Zich and Zichzelf

For accidental reflexive verbs in general, the use of *zich* was more frequent than *zichzelf*. We find 163K (84%) occurrences of *zich* vs. 31K (16%) occurrences of *zichzelf*. For more detailed observations, we restrict attention to verb+subcategorization pairs, that occur at least 50 times in the corpus, and at least 10 times with a reflexive (899 cases, of which, according to the grammar, 163 are inherent reflexive verbs, and 736 are accidental reflexive verbs). Although *zichzelf* in general is rare, we find that 6% of the accidental reflexive verbs (44 of 736) , when used reflexively, occur with a strong reflexive more than 95% of the time. Examples are *zichzelf in*

[2]Manual inspection of a sample suggests that in all uses of *zichzelf bedruipen* involve the *support oneself* meaning.

de weg zitten (hinder oneself), toespreken (address), opvoeren als (present), afschrijven (write off), and *onderbreken (interrupt)*. 34% of the accidental reflexive verbs (247) occur with a strong reflexive more than 50% of the time. 25% of the accidental reflexive verbs (187) occur with a strong reflexive less than 8% of the time. Some examples of the latter group are *beheersen (withhold), voorstellen (introduce), manoeuvreren (manoevre), uitleveren (hand over to), bevrijden (liberate), wassen (wash), (dress), scheren (shave), beschikbaar stellen (make available)*. We do find a number of 'outward directed' verbs among the group of verbs with a strong preference for *zichzelf*, and a number of 'self directed' verbs in the group with a dispreference for *zichzelf*. This is in line with Haspelmath's semantic characterization of such verbs.

The 44 verbs with a strong preference for the strong reflexive *zichzelf* were used non-reflexively 97.1% of the time. The 247 verbs used more often with a strong reflexive than with a weak reflexive were used non-reflexively 95.1% of the time. The 187 verbs used with a strong reflexive less than 8% of the time were used non-reflexively 72.0% of the time. This suggests that there is indeed a relationship between preference for the strong reflexive form and a high relative frequency of non-reflexive use.

Traditionally, it is claimed that inherent reflexives never occur with the strong reflexive *zichzelf*. We can examine empirically whether or not this is in fact true. Of the 163 reflexive verbs in our data-set, 112 (68.7%) occur with *zich* more than 99% of the time (often with only 1 or 2 occurrences of *zichzelf*).

The remaining 51 reflexive verbs occurred with strong reflexive objects more frequently. Here are a number of examples:

(5) a. Nederland moet stoppen zichzelf op de borst te slaan
The Netherlands must stop beating itself on the chest
b. Hunze wil zichzelf niet al te zeer op de borst kloppen
Hunze doesn't want to knock itself on the chest too much
c. Ze verloren zichzelf soms in tactische varianten
They lost themselves in tactical variants
d. Hij verbeeldt zichzelf oogcontact te hebben
He imagines himself to have eye contact

The idiomatic expression *zich/zichzelf op de borst kloppen* (*to boast*) occurs with a strong reflexive 47 times (30% of the time). A few other idiomatic expressions behave similarly. One explanation might be that the idiomatic readings are still transparently linked to the non-idiomatic, accidental reflexive, reading, leading to a certain amount of interference between the two uses.

verb	nonrefl		refl		*zich*		*zichzelf*	
	#	%	#	%	#	%	#	%
straf (*to punish*)	1060	95.7	47	4.3	2	4.2	45	95.8
bescherm (*to protect*)	4921	96.4	186	7.6	95	51.1	91	48.9
vastketenen (*to chain*)	24	34.8	45	65.2	43	95.6	2	4.4

Table 1: Counts and percentages for nonreflexive and reflexive use, and use of weak and strong reflexive pronouns.

5 Statistical Analysis

We used linear regression to determine to what extent there is a correlation between reflexive use of a (non-inherent reflexive) verb and the relative preference for a weak or strong reflexive pronoun.

The data we are dealing with has the form shown in table 1. Establishing a correlation between the percentage of nonreflexive use and the percentage of occurrences of the strong reflexive *zichzelf* with the verb is problematic because the distribution of the percentage of nonreflexive use is far from normal. This is illustrated in figure 1 (left), which shows the percentages in sorted order.[3] A better alternative is to use the ratio of nonreflexive over reflexive use, and the ratio of strong reflexive use over weak reflexive use, and take the log values of these. For nonreflexive use, this gives the distribution in the right pane of figure 1, which is more evenly spread out.

As before, we limit our analysis to verbs that occur at least 10 times with a reflexive meaning and at least 50 times in total, distinguishing uses by subcategorization frames. Figure 2 (left pane) plots the ratio of nonreflexive use over reflexive use (x-axis) against the ratio of strong reflexive forms over weak reflexive forms (y-axis) for all objects. Linear regression (shown as the solid line in fig. 2) gives an r^2 correlation coefficient of 0.162 (statistically significant at $p < 0.001$), with a standard error of 2.07. This means that the ratio of nonreflexive over reflexive use accounts for 16% of the variance in the ratio of strong reflexive over weak reflexive use.

If we count as non-reflexive uses only cases where a verb occurs with a pronoun (as suggested by Haspelmath), 594 verbs remain with frequencies above the cut-offs we used. Linear regression over this data set gives an r^2 of 0.293, and a slightly lower standard error (1.98). If we only consider third person personal pronouns

[3] Statistical analysis was done with R (`http://cran.at.r-project.org`), following the techniques described in Baayen (2008).

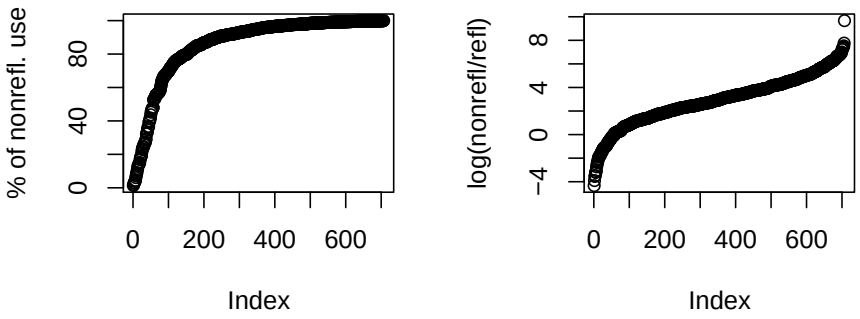


Figure 1: Distribution of percentage of nonreflexive use and ratio of nonreflexive over reflexive use

only (*hem (him), haar (her), hen (them)* and *ze (them)*), 500 verbs remain. We now obtain the result given in fig. 2 (right pane), with an r^2 of 0.332 and a standard error of 1.97.

These results are in line with the findings in Hendriks, Spenader, and Smits (2008). They also observed that restricting object counts to personal pronouns gives a better result than counting all NP-objects. However, for the 32 verbs for which they collected data, they obtain an r^2 of 0.456. As we obtain an r^2 of 0.332, the question arises what might explain this difference. We extracted all verbs from the data-set for personal pronouns that were also used in Hendriks, Spenader, and Smits (2008). 24 of these verbs were sufficiently frequent in our data-set. Linear regression over this limited set gives an r^2 of 0.547 and a standard error of 1.7. One reason for the higher score (compared to Hendriks *et al.*) might be the fact that we take subcategorization frames into account. Another reason might be our use of different frequency cut-offs. What the result also shows, is that our method of data collection in itself does not introduce more noise than the method in Hendriks, Spenader, and Smits (2008). The fact that we obtain a lower score on the larger set of verbs could be due to the fact that the 32 verbs used by Hendriks, Spenader, and Smits (2008) were collected from examples used in the literature. Apparently, these verbs are particularly suitable for demonstrating the statistical correlation to be investigated. Once one takes the full set of verbs into account, however, a fair number of outliers are added as well.

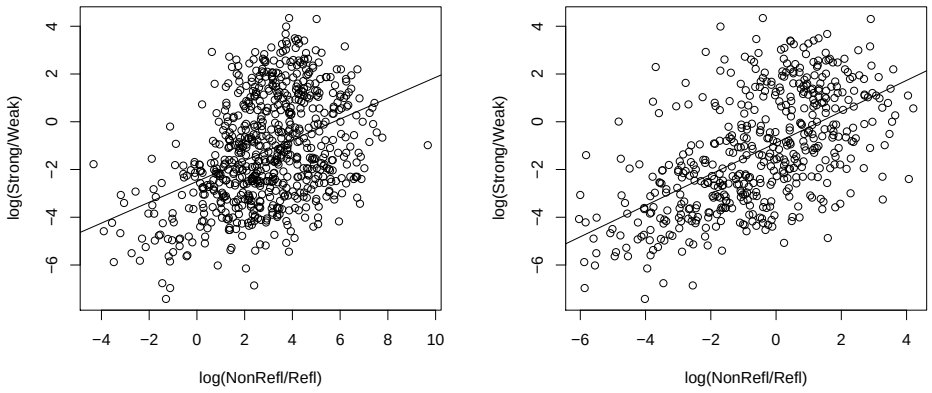


Figure 2: Nonreflexive vs reflexive use compared with strong reflexive over weak reflexive use counting all NP-objects (left) and counting only pronouns (right).

6 Discussion

One of the major ways in which this work tries to improve upon earlier work is by using more data, looking at more verbs (hundreds rather than 30-50) and by using better data (by distinguishing verbs by their subcategorization frames). The assumption is that more data will lead to a better model, and will compensate for irregularities introduced by the fully automated process. Looking at more data did lead to higher correlations for each of the data collection methods, though this effect is not distinguishable from the effect of separating verbs by subcategorization frame.

But looking at more verbs did not give higher correlations. The highest correlation was obtained with the verbs studied by Hendriks, Spenader, and Smits (2008). These are verbs that routinely appear in the literature as good examples of accidental reflexives. One explanation is that these verbs are relatively frequent (although not necessarily frequent in our corpus), and that frequent verbs are the ones for which a speaker may have an expectation of self-directedness or other-directedness. Another explanation is that these verbs in particular might have relatively few different senses, or that they are overwhelmingly used with a sense that has the potential to be both self- or other-directed.

It is still not clear why the ratio of pronominal objects to reflexive objects predicts so much better than taking all objects into account. There are two possible explanations. First, it may be that this restriction in a way also filters out uses

	zichzelf	*zich*		*zichzelf*	*zich*
alleen (only)	109	1	*nu (now)*	16	1
ook (also)	214	9	*wel (certainly)*	14	0
niet (not)	30	9	*min of meer (more or less)*	21	0
slechts (only)	2	0	*alleen maar (only)*	13	1
zelfs (even)	7	0	*zo (that way)*	12	0

Table 2: Choice of reflexive immediately following focus particles

of verbs with senses that essentially cannot be used reflexively. By only counting pronominal objects as non-reflexive objects, the sense of the verb has to be one where the action can be performed on another agent. This would lead to more accurate data (though less data) and may be responsible for the better results.

The other explanation comes from theoretical syntax, Principle A and B of the Binding Theory (Chomsky, 1981) suggests that personal pronouns and reflexives are in complementary distribution when the subject and the object are both animate. In other words, there is a potential for reflexive action only in the case of an animate subject. This means that the ratio for a given action to be self- or other-directed is only reliable if we limited our counts to cases where the subject and object are both animate.

Strictly speaking, comparing the ratio of pronominal objects to reflexive objects doesn't actually give us the ratio of self- vs. other-directed events. This is because we also potentially count cases where the subject is inanimate and the object is a personal pronoun. However, the few corpus studies of grammatical role and animacy that have been done show that the combination of an inanimate subjects with an animate objects is disprefered. Bouma (2008) gives results for spoken Dutch with data for 2,345 sentences from the *Corpus Gesproken Nederlands*. 243 of the sentences had animate objects but among these only 8 (or 3%) occurred with an inanimate subject. Using data from written texts, Øvrelid (2004) looked at 1,000 randomly sampled sentences from the Oslo corpus of Norwegian. 98 of the 1,000 sentences studied had animate objects and of these only 24 had an inanimate subject (24%).

Still, we are able to account for between 30-53% of the data (depending on what dataset is used) using only one predictive factor: how frequently the verb is used with a reflexive object. However, it is also clear that other factors play a role in choosing between a strong and reflexive form. Only strong reflexives can be coordinated, fronted and phonetically focused. This suggests we should take such additional factors into account as well. But coordination of reflexives is rare, and focus or phonetic stress is hard to determine automatically. In a limited number

of cases, one might try to determine focus by taking the preceding expression into account. If the word preceding the reflexive object is a focusing particle, we expect the reflexive following to be *zichzelf*. Table 2 shows that this is indeed the case for a number of expressions that associate with focus.

Factors such as position in the sentence could also be checked. For example, we expect only strong reflexives to be fronted, so we would expect more strong reflexives in initial sentence position. Further, because only strong reflexives can receive sentential accent we would also expect strong reflexives to occur sentence finally more often than weak reflexives (with accidental reflexive verbs). It would be interesting to collect data for the (relative) sentence position of the reflexive (i.e. distance (in words or constituents) from the governing verb or end of the sentence), and to investigate whether a correlation can be found between position and reflexive choice. Geurts (2004) suggests yet another factor. Even non-reflexive verbs like *toedienen* (*to inject oneself*) can use *zich* if the context makes clear the action is a habitual event. This suggests that the presence of temporal adverbs indicating frequency could also play a role. If we can find methods to collect the relevant data automatically, it would be interesting to incorporate them in a multivariate analysis in future work.

Acknowledgements

Jennifer Spenader's work was supported by grant 016.064.062 from the Netherlands Organisation for Scientific Research (NWO).

References

Baayen, R.H. 2008. *Analyzing Linguistic Data*. Cambridge University Press.

Bouma, Gerlof 2008. Starting a Sentence in Dutch. A corpus study of subject- and object-fronting. Groningen Dissertations in Linguistics, 66.

Bouma, Gosse and Geert Kloosterman. 2007. Mining syntactically annotated corpora using xquery. In Branimir Boguraev and Nancy Ide et al., editors, *Proceedings of the Linguistic Annotation Workshop (ACL 07)*, Prague.

Chomsky, Noam 1981. Lectures on Government and Binding. Foris, Dordrecht

Geurts, Bart. 2004. Weak and strong reflexives in dutch. In *Proceedings of the ESSLLI workshop on semantic approaches to binding theory*, Nancy, France.

Haspelmath, Martin. 2004. A frequentist explanation of some universals of reflexive marking. Draft of a paper presented at the Workshop on Reciprocals and Reflexives, Berlin.

Hendriks, Petra, Jennifer Spenader, and Erik-Jan Smits. 2008. Frequency-based constraints on reflexive forms in dutch. In *Proceedings of the 5th International Workshop on Constraints and Language Processing*, pages 33–47, Roskilde, Denmark.

Ordelman, Roeland, Franciska de Jong, Arjan van Hessen, and Hendri Hondorp. 2007. Twnc: a multifaceted Dutch news corpus. *ELRA Newsletter*, 12(3/4):4–7.

Øvrelid, Lilja. 2004. Disambiguation of syntactic functions in Norwegian: modeling variation in word order interpretations conditioned by animacy and definiteness. In Fred Karlsson, editor, *Proceedings of the 20th Scandinavian Conference of Linguistics*, Helsinki.

Reinhart, Tanya and Eric Reuland. 1993. Reflexivity. *Linguistic Inquiry*, 24:656–720.

Smits, Erik-Jan, Petra Hendriks, and Jennifer Spenader. 2007. Using very large parsed corpora and judgement data to classify verb reflexivity. In Antonio Branco, editor, *Anaphora: Analysis, Algorithms and Applications*, pages 77–93, Berlin. Springer.

van Noord, Gertjan. 2006. At last parsing is now operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*. pages 20–42.

Huge Parsed Corpora in LASSY

Gertjan van Noord
University of Groningen
E-mail: G.J.M.van.Noord@rug.nl

Abstract

One of the goals of the LASSY STEVIN project (Large Scale Syntactic Annotation of written Dutch) is a syntactically annotated (manually verified) corpus of 1 million words. In addition, the full STEVIN reference corpus of 500 million words will be syntactically annotated automatically. In this paper, the potential of such huge treebanks for applications in corpus linguistics, natural language processing and information extraction is illustrated.

1 Introduction

The construction of a 500 million word reference corpus of written Dutch has been identified as one of the priorities in the STEVIN programme. Parts of this corpus will be enriched with verified linguistic annotations. The goal of the LASSY STEVIN project is to extend the amount of syntactically annotated material. At the end of LASSY there will be a manually verified treebank consisting of 1 million words. Perhaps more interestingly, the full 500 million word reference corpus will be syntactically annotated automatically, and the resulting huge treebank will be made available as well. The outcome of LASSY therefore contains two types of treebank: a relatively small treebank of high quality on the one hand, and a huge treebank of lower quality on the other hand.

The availability of small quantity, high quality treebanks for research and development in natural language processing is crucial, in particular for training statistical syntactic analysers or statistical components of syntactic analysers of a more hybrid nature. In addition, small quantity, high quality treebanks are important for evaluation purposes for any kind of automatic syntactic analysers.

It is less obvious whether large quantity, lower quality treebanks are a useful resource. In this paper, we illustrate the expected quality of the automatic annotations, and we provide a number of example studies which illustrate the promise of large quantity, lower quality treebanks in areas such as information extraction,

ontology construction, lexical acquisition, corpus linguistics, and natural language processing itself.

2 Automatically Parsed Treebanks

In LASSY, we use the freely available Alpino parser for Dutch [13]. Alpino is computational analyzer of Dutch which aims at full accurate parsing of unrestricted text, and which incorporates both knowledge-based techniques, such as a HPSG-grammar and -lexicon which are both organized as inheritance networks, as well as corpus-based techniques, for instance for training its POS-tagger and its Maximum Entropy disambiguation component.

Although Alpino is not a dependency grammar in the traditional sense, dependency structures are generated by the lexicon and grammar rules as the value of a dedicated feature `dt`. The dependency structures are based on CGN (Corpus Gesproken Nederlands, Corpus of Spoken Dutch) [7], D-Coi and LASSY [15].

Dependency structures are stored in XML. Advantages of the use of XML include the availability of general purpose search and visualization software. For instance, we exploit XPATH (standard XML query language) to search in large sets of dependency structures, and Xquery to extract information from such large sets of dependency structures.

The output of the parser is evaluated by comparing the generated set of named dependencies for a corpus sentence to the set of named dependencies extracted from the manually annotated treebank. Comparing these sets, we count the number of dependencies that are identical in the generated parse and the stored structure. The concept accuracy (CA) measure indicates the proportion of correct named dependencies.

In order to judge the expected quality of automatically parsed corpora, we list the concept accuracy figures for the manually verified sub-corpora developed in STEVIN D-Coi. For these sub-corpora, we compare the dependency structures produced by Alpino with the manually verified dependency structures. The results are aggregated over the various sub-corpora, because it can be expected that the accuracy of syntactic dependencies differ with respect to domain. As can be observed in the table, most sub-corpora can be parsed with an accuracy well over 80%. Newspaper texts are particularly easy for Alpino, because Alpino is trained on the Alpino treebank, which also consists of newspaper text. The *books* sub-corpus was relatively easy, because the corpus sentences all came from a single children's book. Legal texts constitute the hardest sub-domain in this collection.

material	number of words
XMLWiki Wikipedia 2008	110M
Europarl version 3	38M
TwNC newspaper material	531M
Mediargus newspaper material	1397M
European Medicines Agency	14M

Table 1: Some of the corpora which have been automatically parsed in the context of the Lassy project.

sub-corpus	# sentences	CA%	sub-corpus	# sentences	CA%
newsletters	90	83.17	web sites	2833	86.56
wikipedia	240	88.38	press releases	591	84.46
books	276	92.86	brochures	1796	85.84
flyers	306	87.93	manuals	397	80.26
legal texts	279	76.96	newspaper	2267	91.03
policy docs	1266	84.73	proceedings	339	88.08
reports	1115	88.60	*total*	12637	86.52

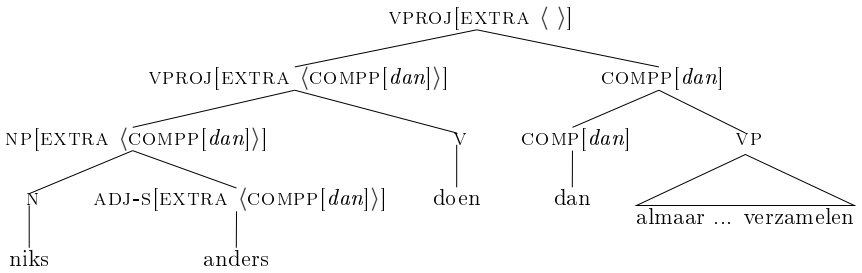
In Lassy, the full STEVIN reference corpus of 500 million words will be automatically annotated. This corpus is not yet available. In the experiments reported in this paper, several other large automatically annotated corpora have been used. Table 1 lists some of the automatically annotated corpora which are currently available.

3 Corpus Linguistics

Huge parsed corpora are useful for research in corpus linguistics. In this section, we will illustrate this by means of an anecdotical example. However, there are also examples of more systematic corpus linguistics research, exploiting huge corpora parsed with Alpino (one such example is presented in Bouma & Spenader, 2009 (this volume)).

In [10], the grammar underlying the Alpino parser is presented in some detail. As an example of how the various specific rules of the grammar interact with the more general principles, the analysis of comparatives and the interaction with generic principles for (right-ward) extraposition is illustrated. In short, comparatives such as comparative adjectives and the adverb `anders` as in the following example, license corresponding comparative phrases (such as phrases headed by `dan` (than)) by means of a feature which percolates according to the extraposition principle.

(1) . . . *niks anders* doen *dan* almaar ruw materiaal verzamelen
. . . nothing else do but always raw material collect
(. . . *do nothing else but collect raw material*)



An anonymous reviewer criticized the analysis, because the extraposition principle would also allow the rightward extraction of comparative phrases licensed by comparatives in topic position. The extraposition principle would have to allow for this in the light of examples such as

(2) De vraag is gerechtvaardigd waarom de regering niets doet
The question is justified why the government nothing does
The question is justified why the goverment does not act

However, the reviewer claimed that comparative phrases cannot be extraposed out of topic, as examples such as the following indicate:

(3) *Lager was de koers nooit dan gisteren
Lower was the rate never than yesterday
The rate was never lower than yesterday

Since the Alpino grammar allows such cases, it is possible to investigate if genuine examples of this type occur in parsed corpora. In order to understand how we can specify a search query for such cases, it is instructive to consider the dependency structure assigned to such examples in figure 1. As can be observed in the dependency graph, the left-right order of nodes does not represent the left-right ordering in the sentence. The word-order of words and phrases is indicated with XML attributes *begin* and *end* (not shown in figure 1) which indicate for each node the begin and end position in the sentence respectively.

The following XPATH query enumerates all examples of extraposition of comparative phrases out of topic. We can then inspect the resulting list to check whether the examples are genuine.

```
//node[@cat="smain" and node[node[@rel="obcomp"]/@end >
          ../node[@rel="hd"]/@begin]/@begin = @begin]
```

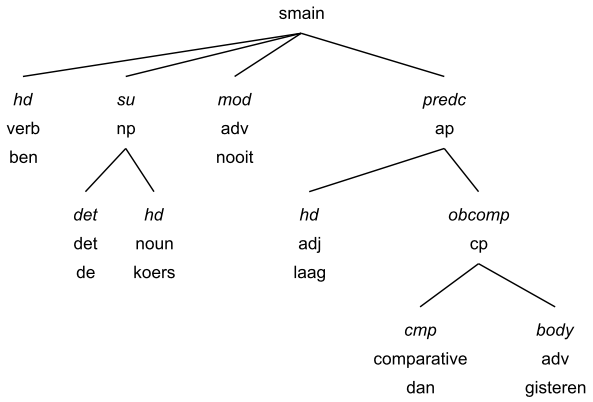


Figure 1: Dependency structure for *Lager was de koers nooit dan gisteren*

The query can be read as: find root sentences in which there is a daughter node, which has a daughter node with relation label `obcomp` (the label used for comparative complements). The daughter node should begin at the same position as the root sentence. Finally, the end position of the `obcomp` node must be larger than the end position of the head of the root sentence (i.e. the finite verb).

In addition to many mis-parsed sentences, we found quite a few genuine cases. A mis-parse can for instance occur if a sentence contains two potential licensers for the comparative phrase, as in the following example in which *verder* can be wrongly analysed as a comparative adjective.

(4) Verder wil ik dat mijn backhand even goed wordt als mijn forehand
Further want I that my backhand just-as good becomes as my forehand
Furthermore, I want my backhand to become as good as my forehand

Some typical, genuine, examples are listed below. It is striking that many examples involve the comparative adjectives *liever* and *eerder*. Also, the list involves examples where adverbials such as *zo, zozeer, zoveel* are related with an extraposed subordinate sentence headed by *dat* which according to the annotation guidelines are also treated as comparative complements.

(5) a. Liever betaalden werkgevers een (hoge) verzekeringspremie , dan opgescheept te zitten met niet volwaardig functionerende medewerkers (Algemeen Dagblad, January 15, 1999)
b. Nooit eerder waren zoveel van de waterlelieschilderijen bijeen op één tentoonstelling als nu in Londen (Algemeen Dagblad, January 27, 1999)
c. Liever huppelt ons land over het hoogpolige tapijt van de VN dan kennis te

nemen van de realiteit in het oerwoud en op de savanne van Afrika (Algemeen Dagblad, September 25, 1999)

d. Beter is het te zorgen dat ziekenhuizen hun verplichtingen volgens de. huidige BOPZ gaan nakomen , dan de rechten van patiënten nog verder aan te tasten (Algemeen Dagblad, August 18, 2001)

e. Dus wat anders konden de LPF'ers de afgelopen week dan zich stil houden (Volkskrant June 1, 2002)

f. Zozeer drukte de 300.000 mensen tellende Chinese gemeenschap haar stempel op de stad , dat zij de bijnaam ' Hongcouver ' kreeg (NRC, August 23)

g. Hetzelfde moet gebeuren als wanneer een man een mooie vrouw ziet , of andersom natuurlijk (Algemeen Dagblad, December 29, 2001)

The examples show that at least in some cases, the possibility of extraposition of comparative phrases from topic must be allowed.[1] More importantly for the purpose of this paper, we hope that the example illustrates that huge parsed corpora are helpful to find new empirical evidence for fairly complicated and suble linguistic issues.

4 Learning Similar Words

The distributional similarity hypothesis is that words which occur in the same context have similar meanings [6]. In order to learn which words have similar meanings, concrete similarity measures have been applied to pairs of weighted context feature vectors, where each vector represents the context of a word, as observed in large text collections. An overview of such methods is given in [5]. The results have been shown to be practically useful for Question Answering, Information Extraction, Paraphrasing, etc.

Many of earlier distributional similarity methods used a *bag of words* to describe the context of words, but in more recent years, approaches where the context of a word is defined by reference to syntactic structure have become more common, and in many cases perform better than the earlier methods [9].

For Dutch, Van der Plas and Bouma [12, 11] compared the performance of various commonly used vector-based methods. In their approach, context features

[1] The examples sometimes appear to indicate a somewhat old-fashioned, poetic style, perhaps explained by the following biblical citation, which also employs the syntactic phenomenon discussed here:

(i) eerder gaat zoo'n kameel door het oog van een naald, dan dat een rijke in zou gaan in het koninkrijk der hemelen

for nouns are defined with respect to the syntactic dependency the noun occurs in. Typical features include 'occurs as the direct object of the verb *to paint*', 'occurs as an apposition to the noun *bag*' and 'occurs in a coordination with the noun *body*'. Each noun is represented by a vector, where each cell indicates the weighted score of a particular context feature. This score could be the raw frequency of the feature found in a large parsed corpus, but the authors show that pointwise mutual information [2] provides a more informative score.

Once all nouns are represented by vectors, a similarity measure is used to compute the similarity of two nouns. Van der Plas and Bouma use a particularly simple method, referred to as Dice-dagger, and introduced for a similar purpose in [3]. In order to compare two vectors v,w, Dice-dagger is defined as:

$$\sum_i 2 \cdot \frac{min(v_i, w_i)}{v_i + w_i}$$

In a web-demo at `http://www.let.rug.nl/~gosse/SetsTwNC` the results of this method can be inspected. For this demo the TwNC newspaper corpus has been parsed with Alpino. For each word the demo returns those words which are closest with respect to the Dice-dagger metric. The method is best appreciated by looking at a number of examples. The method works very well for proper names:

Porsche Audi, Mercedes, BMW, Volkswagen, Volvo, Bentley, Renault, Skoda, Toyota, Saab, VW, Jaguar, Opel, Citroën, Fiat, Peugeot, Honda, Rolls-Royce, Bugatti

Tokyo Tokio, Montreal, Stockholm, Seoul, Kopenhagen, Boston, Melbourne, Genève, Buenos Aires, Zürich, Warschau, Oslo, Bangkok, Lissabon, Boedapest, San Francisco, Dresden, St. Petersburg, Praag

Groningen Eindhoven, Utrecht, Nijmegen, Zwolle, Tilburg, Den Bosch, Haarlem, Arnhem, Leiden, Breda, Leeuwarden, Almere, Amersfoort, Dordrecht, Delft, Maastricht, Lelystad, Enschede, Emmen

Tom Waits Patti Smith, Elvis Costello, Joni Mitchell, Bob Dylan, Brian Wilson, Lou Reed, Paul Simon, Randy Newman, Leonard Cohen, Nick Cave, Stevie Wonder, Frank Zappa, Mick Jagger, Captain Beefheart, David Bowie, Paul McCartney, Frank Boeijen, Van Morrison, Neil Young

Graham Greene Evelyn Waugh, Richard Klinkhamer, Tahar Ben Jelloun, Bram Stoker, David Yallop, Florian Illies, Italo Calvino, George Bernard Shaw, Bert Hiddema, E.M. Forster, Lu Xun, Elias Canetti, Stefan Heym, Gustave Flaubert, Frédéric Beigbeder, Michael Lewis, Anthony Powell, Albert Helman, Kingsley Amis

A number of examples with common nouns:

huis (house) woning, gebouw, pand, kamer, kantoor, boerderij, auto, villa, straat, winkel, tuin, flat, huisje, appartement, hotel, brug, paleis, kerk, ruimte (*house, building, house, room, office, farm, car, villa, street, shop, garden, flat, small house, appartment, hotel, bridge, palace, church, space*)

verliefdheid (enamour, love) heimwee, hartstocht, jaloezie, ontroering, erotiek, liefdesrelatie, fascinatie, vriendschap, begeerte, genegenheid, ontrouw, eenzaamheid, overspel, seksualiteit, gekte, verlegenheid, hysterie, waanzin, schuldgevoel (*homesickness, passion, jealousy, emotion, erotics, love relation, fascination, friendship, lust,love, infidelity, loneliness, infidelity, sexuality, madness, shyness, hysteria, madness, feelings of guilt*)

prei (leek) paprika, ui, bleekselderij, champignon, biet, spinazie, broccoli, selderij, kool, courgette, peterselie, bloemkool, knoflook, komkommer, venkel, tomaat, andijvie, aubergine, wortel (*pepper, onion, blanched celery, mushroom, beet, spinach, broccoli, celery, cabbage, zucchini, parsley, cauliflower, garlic, cucumber, fennel, tomato, endive, aubergine, carrot*)

In her thesis, Van der Plas evaluates the method using EuroWordNet as the target. If words are similar using the automatic method, then these words should also be similar in EuroWordNet. To compute similarity of words in EuroWordNet, the Wu and Palmer metric is used. In a first comparison, she shows that the syntax-based model performs much better than the word-based proximity model for this task.

She shows furthermore that it is important that huge annotated corpora are available, by comparing the result of an experiment which had access to 80 million words, with an experiment which had access to 500 million words. She provides scores for three classes of nouns, depending on frequency. For high frequency nouns the EuroWordNet score increases from .747 to .765; for middle frequency nouns the EuroWordNet score increases from .644 to .737. For low frequency nouns the increase in performance is largest: .488 vs. .666.

These experiments indicate not only that syntactically annotated corpora are an important resource for learning word similarity, but also that the size of such syntactically annotated corpora must be really huge if we also want to apply such techniques successfully for less frequently occurring words.

5 Learning Selection Restrictions

As a final example of the use of huge parsed corpora, we report on a study which shows that the incorporation of automatically learned selection restrictions im-

proves disambiguation performance of the Alpino parser.

Although parsing has improved enormously over the last few years, even the most successful parsers make very silly, sometimes embarassing, mistakes. As a simple example consider the ambiguous Dutch sentence

(6) Melk drinkt de baby niet
Milk drinks the baby not
The baby doesn't drink milk

The disambiguation model of Alpino employs a wide variety of features. In particular, the model also includes features which encode whether or not the subject or the object is fronted in a parse. Since subjects, in general, are fronted much more frequently than objects, the model has learned to prefer readings in which the fronted constituent is analysed as the subject. Although the model also contains features to distinguish whether e.g. `milk` occurs as the subject or the object of `drink`, the model has not learned a preference for either of these features, since there were no sentences in the training data that involved both these two words.

In fact, in about 200 sentences of a parsed corpus of 27 million sentences `milk` is the head of the direct object of the verb `drink`. Suppose that an automatic learning procedure would need at least perhaps 5 to 10 sentences in order to be able to learn the specific preference between `milk` and `drink`. The implication is that we would need a (manually labeled!) training corpus of approximately 1 million sentences (20 million words). In contrast, the disambiguation model of the Dutch parser we are reporting on in this paper is trained on a manually labeled corpus of slightly over 7,000 sentences (145,000 words). It appears that semi-supervised or un-supervised methods are required here.

Note that the problem not only occurs for artificial examples such as (6); here are a few mis-parsed examples actually encountered:

(7) a. Campari moet **u** gedronken hebben
Campari must you drunk have
Campari must have drunk you / You must have drunk Campari
b. De paus heeft **tweehonderd daklozen** te eten gehad
The pope has two-hundred homeless-people to eat had
The pope had two hundred homeless people for dinner

In a recent paper, van Noord [14] describes a technique to include automatically learned lexical preferences in a maximum entropy model, using a method proposed in [8]. Preferences are expressed using pointwise mutual information association scores [4, 2]. The association scores are estimated using an automatically parsed corpus of 27 million sentences.

As illustration, consider the highest scoring verb/direct object pairs: *bijltje gooi_neer, duimschroef draai_aan, goes by time, kostje scharrel, peentje zweet, traantje pink_weg, boontje dop, centje verdien_bij, champagne_fles ontkurk, dorst les, fikkie stook, gal spuw, garen spin, geld_kraan draai_dicht, graantje pik_mee, krediet_kraan, draai_dicht, kruis_band scheur_af, kruit verschiet, olie_kraan draai_open, onderspit delf, oven_schaal vet_in, pijp_steel regen,*

We furthermore list the highest scoring objects of *drink*: *biertje, borreltje, glaasje, pilsje, pintje, pint, wijntje, alcohol, bier, borrel, cappuccino, champage, chocolademelk, cola, espresso, koffie, kopje, limonade, liter, pils, slok, vruchtensap, whisky, wodka, cocktail, drankje, druppel, frisdrank, glas, jenever, liter, melk,*

The paper argues that lexical affinities are also important for other types of dependency. Highest scoring pairs involving a verb and an adverbial are: *overlangs snijd_door, ten hele dwaal, welig tier, dunnetjes doe_over, omver kegel, on_zedelijk betast, stief_moederlijk bedeel, stierlijk verveel, straal loop_voorbij, uitein rafel, aaneen smeed, bestraf spreek_toe, cum laude studeer_af, deerlijk vergis, des te meer klem, door en door verrot, glad strijk_af, glazig fruit,*

The study provides experimental evidence that the inclusion of this type of lexical knowledge in the parser improves parsing accuracy. On the WR-P-P-H part of the D-Coi corpus (a set of manually verified dependency structures for 2267 sentences from the newspaper Trouw of 2001), the Alpino parser obtained a concept accuracy of 90.32%. After re-training the disambiguation model with the inclusion of lexical preferences, the accuracy went up to 90.73%. From this result, we are confident to conclude that huge parsed corpora are a useful resource. We can even use that resource to improve upon the parser that produced this resource in the first place!

Acknowledgement. The Lassy project is carried out within the STEVIN programme which is funded by the Dutch and Flemish governments `http://taalunieversum.org/taal/technologie/stevin`

References

[1] Gosse Bouma and Jennifer Spenader. Gosse bouma and jennifer spenader. the distribution of weak and strong object reflexives in dutch. In Frank van Eynde, Anette Frank, and Koenraad De Smedt, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories*, Utrecht, 2009. LOT Netherlands Graduate School of Linguistics. This volume.

[2] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[3] James R. Curran and Marc Moens. Improvements in automatic thesaurus extraction. In *In Proceedings of the Workshop on Unsupervised Lexical Acquisition*, pages 59–67, 2002.

[4] Robert Mario Fano. *Transmission of Information: A Statistical Theory of Communications*. MIT Press, Cambridge, MA, 1961.

[5] Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 107–114, Ann Arbor, June 2005.

[6] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[7] Heleen Hoekstra, Michael Moortgat, Bram Renmans, Machteld Schouppe, Ineke Schuurman, and Ton van der Wouden. *CGN Syntactische Annotatie*, December 2003.

[8] Mark Johnson and Stefan Riezler. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 154–161, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[10] Leonoor van der Beek, Gosse Bouma, and Gertjan van Noord. Een brede computationele grammatica voor het Nederlands. *Nederlandse Taalkunde*, 7(4):353–374, 2002.

[11] Lonneke van der Plas. *Automatic Lexicon-semantic Acquisition for Question Answering*. PhD thesis, University of Groningen, 2008.

[12] Lonneke van der Plas and Gosse Bouma. Syntactic contexts for finding semantically similar words. In Ton van der Wouden, Michaela Poss, Hilke Reckman, and Crit Cremers, editors, *Computational Linguistics in the Netherlands 2004*, pages 173–186. LOT, 2005.

[13] Gertjan van Noord. **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, pages 20–42, Leuven, 2006.

[14] Gertjan van Noord. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the International Workshop on Parsing Technology (IWPT)*, ACL 2007 Workshop, pages 1–10, Prague, 2007. Association for Computational Linguistics, ACL.

[15] Gertjan van Noord, Ineke Schuurman, and Vincent Vandeghinste. Syntactic annotation of large corpora in STEVIN. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, 2006.

LFG PARSEBANKER: A Toolkit for Building and Searching a Treebank as a Parsed Corpus

Victoria Rosén, Paul Meurer and Koenraad De Smedt
University of Bergen and Unifob AKSIS
LaMoRe Research Group
E-mail: {victoria|paul.meurer|desmedt}@uib.no

Abstract

We present the LFG PARSEBANKER, a comprehensive toolkit for interactive incremental construction of a treebank as a parsed corpus. This web-based toolkit offers an environment for batch and interactive parsing, versioning, inspection of structures, discriminant-based disambiguation, and statistics. It has recently been extended with a structural search facility.

1 Overview

In the context of the TREPIL project we have developed methods and tools for the semi-automatic construction of treebanks as parsed corpora. Our approach to treebanking aims for efficiency by combining automatic parsing and computer-assisted manual disambiguation. Parsing is implemented in XLE [5] into which any compatible LFG grammar [1] can be uploaded. Through the automatic identification of discriminants [3], the manual work can remain focused on simple choices even if annotations can be detailed and complex. This method has been successfully applied in earlier research contexts [11, 6], while we have adapted the use of discriminants to LFG structures [9, 7, 8, 10]. We now focus on the toolkit that implements this method.

A practical and efficient implementation of parsebanking with discriminants presupposes an interactive environment that integrates the manual work with automated services such as treebank navigation, computation of discriminants, structural search, versioning, etc. Extending our earlier work [7], we present an updated version of the LFG PARSEBANKER, a comprehensive toolkit that provides this functionality. The toolkit can be used through a web browser and offers the following webpage views:

1. A *treebank overview* page provides corpus selection and navigation in the selected corpus. A partial screenshot from the overview page is shown in figure 1. This view lists all sentences in the corpus together with information such as the sentence ID, the number of parse solutions, the number of discriminants chosen out of the number of total discriminants, the number of chosen analyses, and the sentence length. The annotator may choose to display additional information; among the possibilities are parse time, version information, whether the annotation process is finished, and annotator comments. This view also offers sorting of sentences depending on their various properties.

Id	*Solutions*	*Disc.*	*Chosen*	*Words*	*Sentence*
1	4	1/44	1	8	**Sofie Amundsen var på vei hjem fra skolen.**
2	6	2/114	1	9	**Det første stykket hadde hun gått sammen med Jorunn.**
3	12	3/201	1	5	**De hadde snakket om roboter.**
4	2+14	1/26	1	11	**Jorunn hadde ment at menneskets hjerne var som en komplisert datamaskin.**
5	7	2/183	1	10	**Sofie var ikke helt sikker på om hun var enig.**
6	64	6/256	1	10	**Et menneske måtte da være noe mer enn en maskin?**
7	16	4/178	1	8	**Ved det store matsenteret hadde de skilt lag.**
8	30+30	3/91	1	17	**Sofie bodde i enden av en vidstrakt villabebyggelse og hadde nesten dobbelt så lang skolevei som Jorunn.**

Figure 1: LFG PARSEBANKER overview page

2. XLE-WEB is an interface to the XLE parser on a web page.[1] This interface presents the parse results for a single sentence. It includes a display of packed structures and shows the computed discriminants. The user can disambiguate the sentence by selecting or rejecting discriminants and thereby retaining or rejecting sets of corresponding analyses.

3. The *parsebanking* page offers views and disambiguation as in XLE-WEB, but also additional parsebank management operations, such as subcorpus and grammar selection. This page also contains a search window (see figure 2).

[1] http://decentius.aksis.uib.no/trepil/xle.xml

4. The *discriminant statistics* page presents a frequency list of chosen discriminants for a subcorpus. Each discriminant is listed with its type, the number of times it has been chosen (i.e. marked as good) and the number of times it has been rejected (i.e. marked as bad).

Most of these components are implemented in Common Lisp and use XML, XSLT and Javascript to serve the interface web pages. C-structure trees (and graphs) are drawn using Scalable Vector Graphics (SVG) and MySQL is used to store the parsebank, although the system is also compatible with other databases.

2 LFG SEARCHTOOL

The current version of the LFG PARSEBANKER offers a new advanced search facility that allows the user to search for structural characteristics in both c-structures and f-structures. This search tool is modeled after TIGERSearch[2] [4], which has been reimplemented and extended to cover f-structures, which are not trees, but directed graphs, possibly with structure sharing and cycles. It has a query language that is suitable for c- and f-structures and is more convenient to use than the generic TIGERSearch syntax, to the extent that a graphical interface is not deemed necessary.

In contrast to the TIGERSearch system, whose treebanks and search indices are static, our system allows for dynamic index updating, such that the index always represents the current state of the dynamic treebank. The index data is stored in a database, but in addition, a representation of the index is kept in memory for fast querying.

2.1 The query language

For many common query types it is a problem that they are complicated (or impossible) to express or that they perform unsatisfactorily when using the basic TIGERSearch syntax. Therefore, an abbreviated node description syntax has been devised, and special search language constructs have been implemented that translate to efficient code. Among the new constructs are:

- An abbreviated syntax for c-structure nodes: inner nodes can be specified using the bare node label, whereas leaf nodes are represented by the quoted surface string. For example, query (1) matches all configurations where an NP node dominates a surface node “barna":

[2] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/

(1) NP >∗ “barna"

- A rule operator ‘→’ to search for c-structure configurations:

  (2) NP → N PP

  (3) NP → (N #x:PP) & #x → (. P NP .∗)

  Query (2) matches all NPs having exactly an N and a PP child, in that order. Query (3) matches the same configuration, where in addition the PP node should have a P node and an NP node as second and third children. The left hand side of the rule operator can be a regular expression over node labels.

- A path operator ‘>(...)’ to search for paths through f-structures (coded as regular expressions over attributes). Examples:

  (4) #f >(COMP TNS-ASP TENSE) “past"

  (5) #f_1 >(TOPIC & OBJ) #f_2

  (6) #f_1 >((COMP | XCOMP)+ (OBJ | OBJth)) #f_2

  Example (4) searches for a simple path from an f-structure to an atomic value, whereas (5) shows a query involving structure sharing: the f-structure #f_2 should be the value of both the TOPIC and the OBJ attribute of #f_1. Functional uncertainty can be expressed using the Kleene star or plus operators, as illustrated in (6).

- A projection operator ‘≫’ relating c-structure nodes and f-structures they project to:

  (7) NP ≫ #f

  This query finds all NP nodes and the f-structures they project to. The projection operator can be combined with the path operator:

  (8) #c ≫(OBJ PRED) “vei"

2.2 The query interface

Queries can be input both on the overview page and on the sentence page. Long-running queries may be stopped at any time, and a list of links to matching sentences is updated dynamically while a query runs. Matches can be inspected by clicking on a sentence number in the match list. In the c-structure, matching nodes are highlighted, whereas in the f-structure, matching paths are shown.

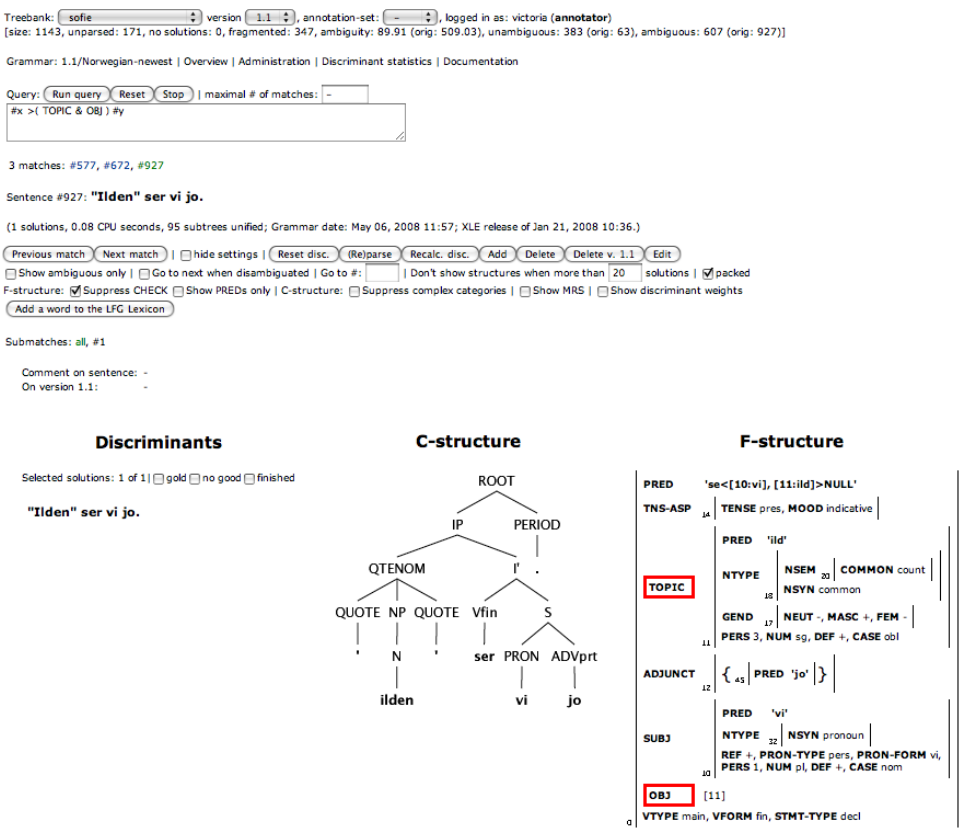


Figure 2: Parsebanking page with query result

The screenshot in figure 2 illustrates a view of the parsebanking page with the results of a search. The treebank "sofie" has been searched with the query in (5) above, which specifies that the same f-structure value must fill both the TOPIC and the OBJ functions, in other words that the sentence has a topicalized object. This treebank has three matches for the query, listed by their sentence ID numbers under the query window. The sentence displayed is *"Ilden" ser vi jo.* ("The fire", we see after all.) The attributes that share the same value are highlighted in the f-structure display.

The toolkit is language independent and can be used with any LFG grammar implemented in XLE, and it has been licensed to several members of the Parallel

Grammar Project [2].

References

[1] Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, Malden, MA, 2001.

[2] Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The Parallel Grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan*, 2002.

[3] David Carter. The TreeBanker: A tool for supervised training of parsed corpora. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, Providence, Rhode Island, 1997.

[4] Wolfgang Lezius. Tigersearch – Ein Suchwerkzeug für Baumbanken. In Stephan Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002), Saarbrücken*, 2002.

[5] John Maxwell and Ronald M. Kaplan. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589, 1993.

[6] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods, a rich and dynamic treebank for HPSG. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories*, pages 117–128. Växjö University Press, 2003.

[7] Victoria Rosén, Koenraad De Smedt, Helge Dyvik, and Paul Meurer. TREPIL: Developing methods and tools for multilevel treebank construction. In Montserrat Civit, Sandra Kübler, and Ma. Antònia Martí, editors, *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 161–172, 2005.

[8] Victoria Rosén, Koenraad De Smedt, and Paul Meurer. Towards a toolkit linking treebanking to grammar development. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 55–66, 2006.

[9] Victoria Rosén, Paul Meurer, and Koenraad De Smedt. Constructing a parsed corpus with a large LFG grammar. In *Proceedings of LFG'05*, pages 371–387. CSLI Publications, 2005.

[10] Victoria Rosén, Paul Meurer, and Koenraad De Smedt. Designing and implementing discriminants for LFG grammars. In Tracy Holloway King and Miriam Butt, editors, *The Proceedings of the LFG '07 Conference*, pages 397–417. CSLI Publications, Stanford, 2007.

[11] Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN) 2001*, Twente University, 2002.

Cultivating Trees: Adding Several Semantic Layers to the Lassy Treebank in SoNaR

Ineke Schuurman
K.U.Leuven
Centrum voor Computerlinguïstiek
ineke.schuurman@ccl.kuleuven.be

Veronique Hoste
University College Ghent
LT3
veronique.hoste@hogent.be

Paola Monachesi
Utrecht University
Uil-OTS
paola.monachesi@phil.uu.nl

1 Introduction

Within the STEVIN[1] project Large Scale Syntactic Annotation of written Dutch (LASSY), a manually corrected treebank of 1 million words is constructed. Lassy is part of a series of annotation projects for modern written and spoken Dutch. More specifically, it is an extension of the D-Coi and CGN projects,[2] and constitutes the core of SoNaR, a 500 million words reference corpus of modern written Dutch.[3] One of the goals of the latter project is to enrich the corrected treebank produced in Lassy[4] with several semantic layers.
For a general overview of the relations between D-Coi, Lassy and SoNaR, cf [19]. In this paper we will concentrate on the semantic layers of SoNaR core: (1) named entity labeling, (2) annotation of co-reference relations, (3) semantic role labeling and (4) annotation of spatial and temporal relations. Of these (2) originates from the STEVIN-project COREA,[5] (3) and (4) from D-Coi, whereas (1) is a new area within STEVIN.

[1]Funded by both the Dutch and Flemish governments, the present joint Dutch-Flemish STEVIN programme was started in 2004 (http://taalunieversum.org/taal/technologie/stevin/)

[2]http://lands.let.ru.nl/projects/d-coi/; http://lands.let.kun.nl/cgn/ehome.htm

[3]http://lands.let.ru.nl/projects/SoNaR/. The first phase of SoNaR started January 2008.

[4]The remaining 499 million words of SoNaR will also be tagged and parsed, but there will be no manual correction.

[5]http://www.cnts.ua.ac.be/~hoste/corea.html

2 Four semantic layers

So far, the creation of semantically annotated corpora has lagged behind dramatically. Within the STEVIN-programme, four birds are now being killed with one stone, with four layers of semantic annotions being added to an existing, manually corrected treebank.
The layers will be added in the order in which they are presented below, this way the layers at the end can profit from the results obtained earlier.

2.1 Named Entity and Co-Reference Labeling

Factoid question answering and information extraction relies heavily on the detection of named entities[6] in texts, since names in general tend to be salient context words. If information is extracted from a text, it often involves answering questions of the type *who, what, where?*, all of which can have names as answers. However, much information about these entities is often hidden in anaphoric references. Because of the high frequency of anaphoric expressions, resolving them is important for text understanding.

Named entity labeling. Lacking substantial Dutch corpora annotated with named entity information we develop annotation guidelines on the basis of a comparative study of existing guidelines such as the MUC[7] [4], ACE[8] and TIDES guidelines[9]. While MUC considered three entity types (person, organization, location), ACE further divides locations into geo-political entities and facilities and also adds weapons, substances, and vehicles. Our main focus is on the TIDES named entity annotation, which consists of entity names, temporal expressions, and number expressions. The expressions to be annotated are "unique identifiers" of entities (persons, locations, organizations), times (dates, times, and durations), and quantities (money, measures, percentages, and cardinal numbers). Several of these are further disambiguated when annotating spatiotemporal expressions, cf. section 2.3. We furthermore investigate the annotation of the metonymic use of named entities, e.g. BMW as organization which can be used to denote a car or an index on the stock market.
On the basis of the semantically enriched, one million word core treebank of SoNaR, at a later stage a combined classifier will be developed in which diverse classifiers (such as a maximum entropy learner, a rule learner and a memory-based

[6] Like names of persons, organizations, and locations or expressions of times and quantities
[7] http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/ne_task.html
[8] http://projects.ldc.upenn.edu/ace/docs/English-Entities-Guidelines_v5.6.1.pdf
[9] http://www.nist.gov/speech/tests/ie-er/er_99/er_99.htm

learner) are combined with different types of output classes (singular class labels versus trigrams) and under different conditions (as in [9]). We will investigate different types of features, including orthographic, syntactic and lexical features and we will include seed-list information. Unannotated data will be used to expand the seed lists (as in [3] and to enhance the instance or rule base [6]. The resulting classifier will be applied to the 499 million word corpus and will provide confidence scores in order to facilitate post-processing.

Co-reference annotation. In order to detect all information available on a given name or entity, co-reference resolution is needed to unveil all the information referring to this given entity. In the last decades, corpus-based techniques have become increasingly popular for the resolution of co-referential relations (e.g. [17], [24] for English). The use of a corpus-based methodology was enabled by the creation of co-referentially annotated corpora, such as those developed within the framework of the MUC-6 and MUC-7 Message Understanding Conferences [11]. A more recent annotation effort is the creation of the English ACE (Automatic Content Extraction)[10] data sets.
In order to allow for the development of a corpus-based system for Dutch, which is able to detect co-referential relations between different types of noun phrases, including named entities, definite and indefinite NPs and pronouns, a substantial annotated corpus is required. This has led to the creation of the KNACK-2002 corpus [13] and the creation of the first machine learning system for Dutch co-reference resolution [12]. The guidelines as used in the KNACK-2002 corpus, which were largely based on the MUC-6 and MUC-7 annotation scheme for English, were the basis for the broader annotation guidelines of the Stevin COREA ("Co-reference Resolution for Extracting Answers") project [10], which is applied in SoNaR. SoNaR will lead to an additional 1 million annotated tokens for Dutch, on top of the 325,000 tokens produced in the projects above.
The following co-referential relations are labeled:

1. **Identity or strict co-reference** as in "**Xavier Malisse** *heeft zich geplaatst voor de halve finale.* **De Vlaamse tennisser** *zal dan tennissen (...)" (English: "Xavier Malisse qualified for the semifinals. The Flemish tennis player will play (...)")*

2. **Time-indexed co-reference** as in "**Bert Degraeve***, die tot voor kort* **gedelegeerd bestuurder van de VRT** *was, gaat aan de slag bij staaldraadproducent Bekaert als* **chief financial and administration manager."** *(English: "Bert Degraeve, until recently managing director of the VRT, starts to work*

[10] www.nist.gov/speech/tests/ace/

at steel wire producer Bekaert as chief financial and administration manager." In the example, the two job descriptions refer to the same person, Bert Degraeve, but at a different point in time.

3. **Type-token co-reference** as in "*Ik verkies* **de rode auto***, maar mijn man wou* **de grijze***". (English: "I prefer the red car, but my husband wanted the gray one.")*. The example sentence talks about two distinct cars, a red one and a gray one. "de grijze" denotes something like an object type rather than an object token.

4. **Part/whole co-reference** as in "*Hij kon zijn auto niet starten. De benzinetank was leeg." (English: "He could not start the car. The gas tank was empty.")*

5. **Modality and negation** as in "**Filip Dewinter** *had wel eens* **de nieuwe burgemeester van Antwerpen** *kunnen worden." (English: "Filip Dewinter could have become the new major of Antwerp.")* The use of a modal verb cluster like "had wel eens kunnen worden" indicates a fuzzy relation between anaphor and antecedent.

6. **Predicate nominals** as in "**Vivendi Universal** *is* **de tweede sterkste stijger binnen de DJ Stoxx50".** *(English: "Vivendi Universal is the second largest performer in the DJ Stoxx50.")* Although justly criticized by [5] and [26], we believe that the annotation of co-referential relations between predicate nominals is useful from an application point of view (e.g. information extraction).

7. **Appositions** as in "**Jones***,* **de voorzitter van de raad van bestuur***, verminderde zijn belang met 0,2%." (English: "Jones, the president of the Board of Directors, reduced its interests by 0.2%")*. A special case of co-reference is that of co-referential appositions. Appositions come in two flavours: repetitive and restrictive.

8. **Bound anaphora** as in "**Niemand** *verliest graag* **zijn** *job." (English: "Nobody likes to lose his job".)* Rather than making statements about singular objects in the real world, the preceding example expresses properties of general categories.

9. **Metonymy** as in "**Laken** *heeft gedurende de hele periode 1960-1961* **zijn** *rol in de coulissen gespeeld." (English: "During the whole period 1960-1961 Laken has played its role behind the scenes.)* in which the castle of king Baudoin, Laken, is a common description of the Belgian monarchy.

In the COREA project, the co-reference annotations were based on the annotated corpora of the D-Coi project, which provide ample syntactic information. This same approach is used in SoNaR. The annotation process is accelerated through the application of the automatic co-reference resolution system developed in the COREA project, based on [12], which is retrained during the annotation process.

2.2 Semantic Role Labeling

During the last few years, corpora enriched with semantic role information have received much attention, since they offer rich data both for empirical investigations in lexical semantics and large-scale lexical acquisition for NLP and Semantic Web applications. Several initiatives have emerged at the international level to develop annotation systems of argument structure. Two projects have a leading position in this area, namely FrameNet [14] and PropBank [15] with their own semantic annotation schemes. Of these two, it was decided in D-Coi to annotate the corpus with semantic roles according to the PropBank approach as it provides fewer, more syntactically driven argument labels than FraMeNet.
In order to annotate part of the D-Coi corpus, the PropBank guidelines have been revised on the basis of the annotation process. During the annotation process, some problems have emerged which we aim to solve in SoNaR:

1. Linguistic problems: during the annotation some phenomena have been encountered for which linguistic research does not provide a standard solution yet. They have been discarded but it now has become imperative for us to address them.

2. Interaction among levels: there are examples in which the annotation provided by the syntactic parser is not correct as in the case of a PP which was labeled as modifier by the syntactic annotation but which should be labeled as argument according to the PropBank guidelines. Furthermore, problems have been attested with respect to PP attachment, that is the syntactic representation gives sometimes correct, sometimes incorrect structures and at the semantic level it is possible to disambiguate.

One advantage of employing PropBank for the annotation of semantic roles is that it is quite suitable for `automatic` semantic role labeling. However, in the case of Dutch, there was no semantically annotated corpus available that could be used as training data. In the D-Coi project, a novel approach to rule-based tagging based on D-Coi dependency trees has been proposed [25]. More specifically, a basic mapping between nodes in a dependency graph and PropBank roles was defined.
The approach is implemented in a rule-based semantic argument tagger, called

XARA (XML-based Automatic Role-labeler for Alpino-trees) [25]. The cornerstone of Xara's rule-based approach is formed by XPath expressions. A rule in XARA consist of an XPath expression that addresses a node in the dependency tree, and a target label for that node, i.e. a rule is a *(path,label)* pair. The evaluation carried out shows that XARA achieves a precision of 65,11%, a recall of 45,83% and an F-score of 53.80. In order to evaluate the labeling of XARA, the output of XARA's semantic role tagger has been compared with the manual corrected annotation of 2,395 sentences. Since rules in XARA cover only a subset of PropBank labels, recall is notably lower than precision.
After a corpus has been tagged automatically by XARA, manual annotation can be performed relatively fast, since annotators only need to correct XARA's output instead of starting annotation from scratch.
The manually corrected sentences have been used as training and test data for a Semantic Role Labeling (SRL) classification system (i.e. Tilburg Memory based learner (TiMBL)). The features employed describe the predicate (stem, voice) and the candidate argument, that is category, dependency label, POS tag, the head word and its POS tag. In comparison to experiments in earlier work, relatively few training data was available in D-Coi: the training corpus consisted of 2,395 sentences which comprise 3066 verbs, 5271 arguments and 3810 modifiers. To overcome the data sparsity problem, the classifier was trained using the leave one out (LOO) method. With this option set, every data item in turn is selected once as a test item, and the classifier is trained on all remaining items. Except for the LOO option, only the default TiMBL settings were used during training, to prevent overfitting because of data sparsity. We refer to [25], [18] for further details.

The classifier obtained a precision of 70.27% a recall of 70.59% and an F-score of 70.43. In the annotation of the SoNar treebank, our goal is to use the classifier to annotate the new set of data and improve in this way its performance.

2.3 Spatiotemporal Labeling

Applications like multidocument, and even multilingual, summarization or question answering relies to a large extent, not only on the detection of temporal and geospatial expressions in texts, but also on their disambiguation. In SoNaR, we locate eventualities on a time axis (the *when* of an eventuality) and on a contemporary, geographical map (the *where*) whenever relevant.
Within D-Coi the development of the MiniSTEx (Mini SpatioTemporal Expressions) annotation scheme for temporal and spatial annotation was started, cf. [22, 21, 23]. The version of the annotation scheme used in SoNaR[11] makes uses of the informa-

[11] In D-Coi, also a general spatial component was developed, which is neglected in SoNaR.

tion contained in the treebank, including the three other semantic layers.[12] Named Entity labeling, for example, is informative in order to determine whether a spatiotemporal expression is used in a literal way or as a metonym (which affects the annotation).

The annotation scheme reflects the state of the art in geospatial and temporal annotation. With respect to the latter, TimeML [20] and TIDES [8] come to mind. Geospatial annotation as such is far less widespread and standardized. Only recently a scheme for geospatial annotation came out: SpatialML [1]. However, the subtask of disambiguation is also a subject in geographic information extraction. Some approaches in this field can be found in [7, 16, 27]. But especially as far as the Netherlands and Belgium are concerned[13] the geospatial component of the MiniSTEx scheme goes into more detail, explaining (verbatim) where something is located, for example stating that *Haren* is part of the municipality of *Borgloon* which is in the province of *Limburg* etc.[14] This way reasoning is advanced as the annotated corpus should be self-contained, i.e. the user should be able to understand a text without having a full spatiotemporal database at his disposal. In the annotation process itself, however, such a database plays a central role, as it contains the common spatiotemporal knowledge of the intended (Flemish/Dutch) audience. Other spatiotemporal knowledge is expected to be contained in the texts, although data obtained during annotation will be used to further populate the database.
With respect to the temporal component, the scheme developed within D-Coi is more detailed than, for example, TimeML in that it makes explicit between which dates *Easter* may fall, or when it was *Easter* in a specific year, taking into account the country and religion under consideration.

Contrary to TimeML and SpatialML, MiniSTEx handles temporal and (geo)-spatial annotation in one go, using a similar approach. It also handles *geotemporal* expressions, i.e. expressions associated with a combination of geospatial and temporal properties (for example to express that between the First and the Second World War *Libya*, nowadays an independent country, was a province of Italy).
Another characteristic of MiniSTEx is that full advantage is taken of the fact that the origin of the texts is known as the metadata contain the date (sometimes even the time) and place of publication, and also the title of the source (newspaper etc). From the latter the `background` of the text can be determined, and thus the `intended audience` of the text can be inferred, which to a large extent deter-

[12] In another project, an implementation is investigated in which no other layers are available but tagging and chunking.

[13] And to a lesser extent also neighbouring countries, and other countries the intended audience of a text (cf. [22]) is expected to be familiar with.

[14] In case the value for one of these inbetween levels is unknown, especially for entities in countries the intended audience is not supposed to be very familiar with, *XX* is used.

mines how a spatiotemporal expression is interpreted, taking into account Grice's maximes which can be paraphrased as "Don't say too much and don't say too little." The most obvious interpretation of a (spatiotemporal) expression often will not be clarified by the author, whereas other interpretations will. This information is used to calculate which interpretation of *Dover* is likely to be meant (UK, USA?), or which date(s) should be associated with, for example, *Thanksgiving* (USA, Canada?) or *summer* (which hemisphere?). In case of *Dover*: which town in which country will be referred to in a specific text does not depend on its size or population [27], but on the intra- and extratextual context in which the name is used. In a Dutch or Belgian text, a plain *Dover* will refer to the city in the UK, not to the capital of Delaware in the USA, although the latter has the larger population. A last characteristic is that quite some attention is paid to (properties with respect to) tense & aspect of the language under consideration (Dutch within SoNaR).

2.4 Apparent contradictions between layers

Labels attributed in the various layers may mesh with each other. For example, an adverb of repetition (like *again* (opnieuw, weer)) will get a label ArgM-TMP to indicate its semantic role. But in general this type of ArgM-TMP does not qualify for a spatiotemporal tag. The same holds for abstract locations (like "in his speech"), which will get a label ArgM-LOC but no (geo)spatial tag. Sometimes they may also mesh with the underlying syntactic annotation. This has to do with the fact that all annotations were developed separately, so their definitions of related concepts may differ to some degree. For example, what is considered an argument/complement and what a modifier differs between syntactic analysis and semantic role labeling. Roughly, the first considers elements to be arguments in case they can not be left out without causing ungrammaticality, whereas in the PropBank approach we adopted "a semantic role is being marked as an argument, if it frequently occurs in a corpus and is specific to a particular class of verbs" [2]. Both approaches will not necessarily lead to the same result. The question is whether this is problematic. Our feeling is that it is not a problem as long as the differences are of a systematic nature and well documented.

2.5 Too much information?

The structures the users are confronted with can be very complex when all types of semantic information are shown. The idea, however, is that although all information is present at the level of the xml-structures, the user can select one of more types of information to be shown in the trees.

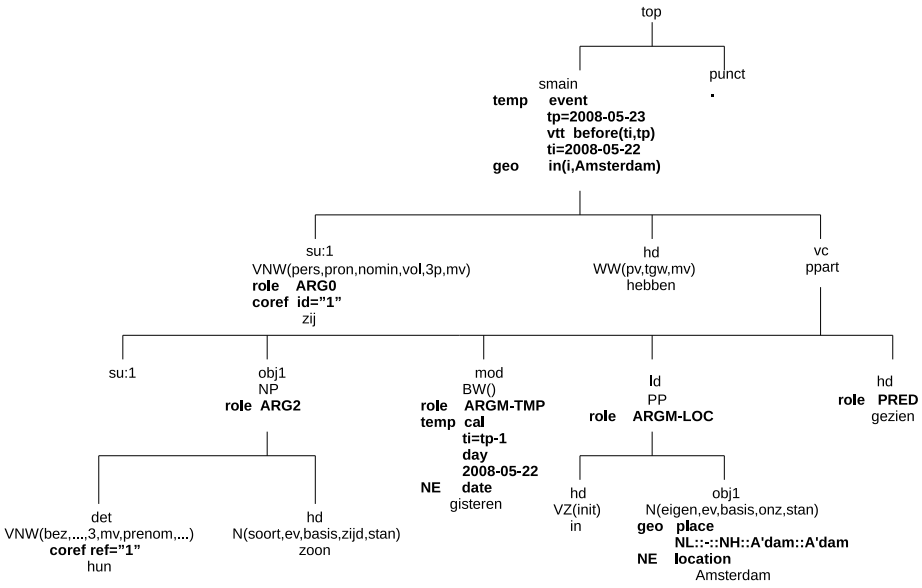


Figure 1: An enriched tree (simplified)

3 Conclusions

With SoNaR a whole series of manually corrected layers of annotation (from part of speech tagging over syntactic analysis to several semantic annotations) are becoming available for the same corpus of 1 million words. In se the same schemes are used in several other (STEVIN) projects, thus, especially when (part of) these data are manually corrected, enlarging the amount of data available for each type of annotation.

With respect to correction, choices were made: have everything corrected by two persons (inter-annotator agreement) or have everything corrected by one person with just parts by more persons (random consistency checks). For semantic role labeling the first option was choosen (smaller corpus), for the other layers the second option. Note that in all cases the (corrected) labels of 'previous' layers will be taken into account (errors will be reported and corrected), but that some relations are stronger than others: whereas, for example, (time-indexed) coreference labeling is essential for spatiotemporal labeling, semantic role labeling is not. Therefore, coreference is taken into account in the MiniSTEx tool, whereas the roles ArgM-LOC and ArgM-TMP are only used of in the correction phase,

References

[1] SpatialML: Annotation Scheme for Marking Spatial Expressions in Natural Language, October 1 2007. MITRE Corporation.

[2] O. Babko-Malaya. *Guidelines for Propbank framers*, September 2005.

[3] S. Buchholz and A. van den Bosch. Integrating seed names and n-grams for a named entity list and classifier. In *Proceedings of LREC-2000*, pages 1215–1221, 2000.

[4] N. Chinchor. MUC-7 Named Entity Task Definition Version 3.5 (web). 1997.

[5] S. Davies, M. Poesio, F. Bruneseaux, and L. Romary. Annotating coreference in dialogues: Proposal for a scheme for MATE. http://www.hcrc.ed.ac.uk/ poesio/MATE/anno_manual.htm, 1998.

[6] F. De Meulder and W. Daelemans. Memory-based named entity recognition using unannotated data. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, pages 208–211, 2003.

[7] J. Ding, L. Gravano, and N. Shivakumar. Computing Geographical Scopes of Web Resources. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, September 10–14 2000.

[8] L. Ferro, L.d Gerber, I. Mani, B. Sundheim, and G. Wilson. *TIDES 2005 Standard for the Annotation of Temporal Expressions*, 2005.

[9] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, volume 168-171, 2003.

[10] I. Hendrickx, G. Bouma, F. Coppens, W. Daelemans, V. Hoste, G. Kloosterman, A.-M. Mineur, J. Van Der Vloet, and J.-L. Verschelde. Coreference Resolution for Extracting Answers for Dutch. In *Proceedings of LREC 2008*, 2008.

[11] L. Hirschman and N. Chinchor. MUC-7 coreference task definition. version 3.0. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[12] V. Hoste and W. Daelemans. Learning Dutch Coreference Resolution. In *Proceedings of CLIN 2004*, pages 133–148, 2004.

[13] V. Hoste and G de Pauw. KNACK-2002: a richly annotated corpus of Dutch written text. In *Proceedings of LREC 2006*, 2006.

[14] C.R. Johnson, C.J. Fillmore, M.R.L. Petruck, C.F. Baker, M.J. Ellsworth, J. Ruppenhofer, and E.J. Wood. *FrameNet: Theory and Practice*, 2002.

[15] P. Kingsbury, M. Palmer, and M. Marcus. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*, San Diego. California, 2002.

[16] J. Leidner. *Toponym Resolution in Text*. PhD thesis, University of Edinburgh, 2007.

[17] J. McCarthy. *A Trainable Approach to Coreference Resolution for Information Extraction*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst MA, 1996.

[18] P. Monachesi, G. Stevens, and J. Trapman. Adding semantic role annotation to a corpus of written Dutch. In *Proceedings of LAW-07*, Prague. Czech Republic, 2007. ACL 2007 workshop.

[19] N. Oostdijk, M. Reynaert, P. Monachesi, G. Van Noord, R. Ordelman, I. Schuurman, and V. Vandeghinste. From D-Coi to SoNaR: A reference corpus for Dutch. In *Proceedings of LREC*, Marrakech, Morocco, 2008.

[20] R. Sauri, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky. *TimeML Annotation Guidelines, version 1.2.1.*, 2006.

[21] I. Schuurman. Spatiotemporal Annotation on Top of an Existing Treebank. In K. De Smedt, J. Hajic, and S. Kuebler, editors, *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories*, pages 151–162, Bergen, Norway, 2007.

[22] I. Schuurman. Which New York, which Monday? The role of background knowledge and intended audience in automatic disambiguation of spatiotemporal expressions. In *Proceedings of CLIN 17*, 2007.

[23] I. Schuurman. Spatiotemporal annotation using MiniSTEx: How to deal with alternative, foreign and obsolete names? In *Proceedings of LREC 2008*, Marrakech, Morocco, 2008.

[24] W.M. Soon, H.T. Ng, and D.C.Y Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

[25] G. Stevens. Automatic Role Labeling in a Dutch Corpus. Master's thesis, Utrecht University, 2006.

[26] K. van Deemter and R. Kibble. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637, 2000.

[27] R. Volz, J. Kleb, and W. Mueller. Towards ontology-based disambiguation of geographical identifiers. In *WWW2007*, Banff, Canada, May 8-12 2007.

Similarity Rules!
Exploring Methods for Ad-Hoc Rule Detection

Markus Dickinson
Indiana University
Department of Linguistics
md7@indiana.edu

Jennifer Foster
Dublin City University
School of Computing
jfoster@computing.dcu.ie

1 Introduction and Motivation

One problem facing the extraction of treebank grammars is that of *ad hoc* rules, rules used for constructions specific to one data set and unlikely to be used on new data (Dickinson, 2008). These rules can be erroneous, cover ungrammatical text, or reveal issues with the treebank's annotation scheme. These are significant problems since training on erroneous data can be detrimental to parsing performance (e.g., Dickinson and Meurers, 2005; Hogan, 2007), and the use of precision grammars in grammar checking and generation requires distinctions between grammatical and ungrammatical sentences (e.g., Bender et al., 2004). Ad hoc rules are especially problematic when they point to inconsistent aspects of the annotation scheme, as the scheme forms the basis of any analysis using it.

Together with these problematic cases, there is also the practical issue of determining which treebank rules are useful for parsing new data: ad hoc rules can also simply be rules that do not generalize. Although frequency is often used to determine generalizability, this is not unproblematic, as low-frequency rules can be valid and potentially useful rules (e.g., Daelemans et al., 1999); high-frequency rules can be erroneous (e.g., Dickinson and Meurers, 2005); and frequency is not completely related to usefulness for parsing (e.g., Foth and Menzel, 2006). Furthermore, infrequent rules in one genre may be quite frequent in another (Sekine, 1997). Thus, methods are needed to determine which rules are generalizable to new data or not.

The previous detection of ad hoc rules, while effective, could be improved in several ways. First, it currently relies on treebank-specific knowledge, such as knowing which part-of-speech categories were *equivalent* (Dickinson, 2008). If

we can justify removing the dependence on corpus-specific properties, the methods can be made more corpus-independent.

Additionally, it is not clear whether the key properties identifying ad hoc rules are based on frequency, some other criterion—namely similarity to other rules—or both. Understanding the contribution of similarity is crucial if we are to move beyond using ad hoc rule detection simply for treebanking and into realms which already use frequency information, such as parsing.

Finally, the methods have only been developed to analyze rules which occurred in some observed data. Although this is useful for examining rule generalizability, it does not tell us anything about the quality of the rules that never occurred in that data. Identifying the quality of unseen rules can begin to point to how one might generalize a grammar to cover new types of data, for cross-genre parsing (cf. Gildea, 2001; McClosky et al., 2006).

We thus examine the role of similarity in ad hoc rule detection and show how previous methods can be made more corpus independent and more generally applicable. After reviewing the previous methods in section 2, we turn to the rationale for similarity-based rule comparison without any treebank-specific knowledge in section 3.1. As we will see, the idea that similarity to other rules indicates the reliability of a rule does not require any corpus-specific information. Building on this, we discuss the component parts of rule comparison, namely frequency and similarity, in section 3.2 and outline how to adapt the methods for unseen rules in section 3.3. We evaluate the methods in section 4, showing that the revised metrics for calculating ad hoc scores are as good as, if not better than, the previous ones, while being more general, providing a foundation for grammar generalization.

2 Background

Dickinson (2008) detects ad hoc rules by first grouping rules into equivalence classes and then looking for similar rules across the grammar. Rules with the same mother are grouped into equivalence classes by the following steps:

1. Remove daughter categories that are always non-predictive to phrase categorization, i.e., always adjuncts, such as punctuation.
2. Group head-equivalent lexical categories, e.g., NN (common noun) and NNS (plural noun).

Then, two methods are used to detect ad hoc rules. The first method (*whole daughters scoring*) directly accounts for similar rules across equivalence classes. Each rule type is assigned a *reliability score* by adding 1 for every rule token within

the equivalence class and adding $\frac{1}{2}$ for every rule token in a highly similar equivalence class. Rules with the lowest scores are flagged as potentially ad hoc.

To determine similarity, a modified Levenshtein distance is used, where only insertions and deletions are allowed; a distance of one qualifies as highly similar. Substitutions are not used, as they are too problematic to include, Consider the erroneous rule VP → RB, which occurs once in the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993). With substitutions, there would be 760 "comparable" instances of VP → VB, despite the vast difference in category (verb [VB] vs. adverb [RB]).

The other method of detecting ad hoc rules (*bigram scoring*) calculates reliability scores by abstracting a rule to its bigrams and examining which bigrams the classes do not have in common. Using equivalence classes in the calculations, both methods effectively identify ad hoc rules. Since the whole daughters method is more effective across a range of tests, we focus on that method in this paper, although our work is applicable to both.

3 Corpus-independent similarity

3.1 Corpus-independence

The notion of similarity above, namely the comparison to whole lists of daughters, is a useful way to compare rules, but this is done only after rules have been reduced into equivalence classes. While the use of equivalence classes is well-motivated—e.g., it generally makes no difference whether a rule has punctuation or not—this is not always ideal.

First, the groupings into head-equivalent categories are not always sound. For example, JJ (adjective) and JJR (comparative adjectives) generally predict the same mother (ADJP) and generally modify nouns. However, there are syntactic differences. As an example, consider (1): comparative adjectives are used in correlative *the*-clauses (Bies et al., 1995, p. 303ff), whereas positive adjectives are generally not. This is a syntactic context where the two categories are not replacable; treating them separately captures this.

(1) The sooner our vans hit the road ... [$_X$ the/DT easier/JJR] it is for ...

Secondly, by comparing rules to similar rules, we are already naturally capturing equivalences among rules. The categories JJ and JJR often are replacable, which we can tell by the observable fact that they behave similarly with respect to other comparable rules. For instance, NP → DT JJ NN (9866 tokens) and NP → DT JJR NN (234 tokens) are comparable rules because they both are one step away

from NP → DT NN (29,217 tokens). Grouping rules around the more basic rule NP → DT NN already indicates similar properties, without requiring pre-specification.

If we can remove the corpus-dependent rules for forming equivalence classes and achieve comparable results, we prefer to do so, in order to be able to work on a variety of corpora without learning the annotation scheme. In this paper, we thus remove the criteria for forming equivalence classes, and calculate reliability scores, using rules as they are. These equivalences, though, still provide a useful goal: Levenshtein distance is effective as a means for determining similarity because it captures properties such as the removability of adjuncts and the natural equivalence of categories.

By more automatically creating equivalences between rules, we are more sensitive to the surrounding categories. For example, a category may always be an adjunct, but it can still be important to know that the adjunct exists within a certain rule. Consider (2a), with the rule ADVP → RB RB -LRB- PP -RRB- PP. Although both -LCB- and -RCB- (codes for left and right curly brackets) are adjuncts, and labeled as such (-LRB-, -RRB-), observing them within another structure is odd, given that the preferred analysis is to embed such bracketed material, as in (2b). We see here that even adjuncts, such as brackets, provide useful information about a rule being ad hoc.

(2) a. . . . they try * to build it [$_{ADVP}$ somewhere/RB else/RB -LCB-/-LRB- [$_{PP}$ in Europe] -RCB-/-RRB- [$_{PP}$ besides the U.K.]] ,

b. [$_{ADVP}$ somewhere/RB else/RB [$_{PRN}$ -LRB- in Europe -RRB-] [$_{PP}$ besides the U.K.]]

3.2 Contributions of information

The methods are based on two major components, frequency of a rule and its similarity to other rules. Removing the equivalence class restriction, as outlined above, we still have the following as our method for computing the whole daughters score, which we use for *reliability scores* in this paper:

1. For every identical rule token, add 1.
2. For every highly similar rule token, add $\frac{1}{2}$.

It is clear from this formulation that we can split the calculation into a frequency score (#1) and a similarity score (#2). Given that frequency is already something which has been used effectively in parsing models, we need to more fully investigate the role that similarity plays in the detection of ad hoc rules, as it has the potential to provide more fine-grained information not available from frequency. When we calculate *similarity scores* on their own, we simply count one

for each similar rule (instead of using $\frac{1}{2}$), as this has a more natural interpretation as the number of similar rules.

3.3 Unseen rules

The method currently only calculates scores for rules which occurred in one data set, namely the training data. But any rule in a set of heldout data can be compared to see how similar it is to the rules in the grammar from the training data. The scoring outlined above can be used, with the only difference that the identical and similar rules are in a different data set. For example, in (3), we have two rules which did not appear in our training data (see section 4); in the case of NP → DT RB PRN S (3a), we have a similarity score of 0, as no rules in the training data are similar, and the rule is incorrect. In the case of the unseen and correct NP → JJ " JJ NN (3b), we have a similarity score of 866, i.e., 866 similar rules.

(3) a. A put option gives its holder [$_{NP}$ the/DT right/RB [$_{PRN}$ but not the obligation] [$_{S}$ * to sell a stock ...]]

b. see [$_{NP}$ various/JJ "/" unmet/JJ needs/NN] , "/" ...

When examining a data set, then, there are two ways one can detect ad hoc rules. On the one hand, one can calculate similarity scores compared to other rules in the same data set. This indicates which rules are inconsistent with other rules in the same grammar and is thus useful for treebanking. On the other hand, one can calculate similarity scores compared to a different data set, indicating which rules are more likely to have emerged from a different grammar. These rules not only might be ad hoc, but they might be indicative of a unique type of construction, whether because the genre is different or because someone else annotated the data.

4 Evaluation

4.1 Unreliability scores

To evaluate, we follow Dickinson (2008) and see how well the scores combining frequency and similarity—the reliability scores—predict a rule's "ungeneralizability." We train on sections 2-21 of the WSJ, and evaluate on section 24. Table 1 shows the ungeneralizability rate for the whole daughters method, for different thresholds.

The results are quite good and dramatically surpass the values presented in Dickinson (2008). For example, a threshold of 50 in the previous whole daughters

Threshold	Rules	Unused	Ungeneralizability
1.0	1625	1617	99.51%
2.0	2801	2785	99.43%
3.0	3515	3479	98.97%
4.0	4011	3965	98.85%
5.0	4412	4357	98.75%

Table 1: New whole dtr. ungeneralizability (WSJ-24)

method with equivalence classes identifies 3548 rules with 96.93% ungeneralizability. For that same approximate number of rules (threshold=3.0), the method without equivalences has 98.97% precision.

Testing across genre Table 2 shows the results of evaluating the whole daughters method on a new set of 1,000 gold standard parse trees (Foster and van Genabith, 2008). The sentences were taken from the British National Corpus (Burnard, 2000), and each was selected on the basis that it contains a verb which does not appear in the WSJ training data. Because there are only 1,000 hand-corrected parse trees, we perform a five-fold cross-validation with training/test splits of 800/200. Again, abandoning the use of equivalence classes appears to be effective. At a threshold of 50.0, the whole daughters method making use of equivalence classes has a precision of 88.51% for 790 rules, whereas, for a similar number of rules (threshold of 5.0), the new method has a precision of 92.52%. At a threshold of 35.0, the old method has a precision of 88.59% for 708 rules, whereas for a threshold of 3.0, the new method has a significantly higher precision of 94.14%.

Threshold	Rules	Unused	Ungeneralizability
1.0	388	376	96.91%
2.0	585	559	95.56%
3.0	683	643	94.14%
4.0	758	707	93.27%
5.0	802	742	92.52%

Table 2: New whole dtr. ungeneralizability (BNC1000, five-fold)

To better see what happens with different genres, we also examined what happens when we train on the original training data (WSJ2-21)[1] and evaluate on the BNC1000. For approximately 1,600 rules (new whole daughters threshold of 1.0,

[1] Since the BNC1000 does not contain traces, these first had to be removed.

old of 8.0), the new method without equivalence classes does a better job (99.25% vs. 98.92%), and for approximately 4,300 rules (new threshold of 5.0, old of 81.0), the new method achieves a precision of 98.66% versus 96.84% for the old method. These results all indicate that the whole daughters method without equivalence classes is superior to the method with them, in terms of predicting which rules are useful for new text. The results are even comparable across genres.

4.2 Similarity vs. Frequency

Based on the previous results, the methods identify ungeneralizable rules quite accurately. The remaining question is: to what extent is this an effect of frequency and to what extent is this an effect of similarity?

Threshold	Rules	Unused	Ungeneralizability
1	8776	8627	98.30%
2	10,741	10,475	97.52%
3	11,601	11,253	97.00%
4	12,131	11,723	96.64%

Table 3: Frequency ungeneralizability (WSJ-24)

Frequency ungeneralizability As we can see in table 3, frequency on its own is a solid indicator of a rule's ungeneralizability, accurately identifying thousands of rules which do not appear in the development data. However, in identifying so many rules, it is rather coarse, not allowing us to sort infrequent but useful rules from infrequent but problematic rules.

Threshold	Rules	Unused	Ungeneralizability
0	1851	1819	98.27%
1	2622	2571	98.05%
2	3147	3080	97.87%
4	3865	3769	97.52%

Table 4: Whole dtrs. similarity (WSJ-24)

Similarity ungeneralizability For the similarity scores (i.e., number of similar rules without including the frequency of the rule in question), we can see in table 4 that whole daughters scoring is also effective at identifying ungeneralizable rules.

More than that, this scoring is providing information more fine-grained than that available from frequency. Most of the 32 rules with 0 scores which appear in the new data are fairly frequent (9 occur more than 100 times in the training data and are thus clearly reliable), which is why the reliability scores, combining frequency and similarity, work best.

Consider also that 1625 of the 1851 identified rules are single-occurrence rules. Of these, 1617 (99.51%) do not generalize to the development data. In fact, for all rules with frequencies of 1, if the similarity score is 2 or less, the ungeneralizability rate is 99.48% (2661/2675), and for similarity scores of 10 or less, it is 99.17% (4085/4119), higher than the 98.30% for all single-occurrence rules (table 3). It thus seems that, for low-frequency rules, similarity scores can sort rules within their frequency class.

4.2.1 Unseen rules

Taking into account rules in the evaluation data which did not occur at all in the training data, we can more fully evaluate how well similarity scores extend to new data. Table 5 shows how the scores change, once unseen rules are also included in the evaluation, dropping dramatically (cf. table 4).

Threshold	Rules	Unused	Ungeneralizability
0	1952	1819	93.19%
1	2747	2571	93.59%
2	3290	3080	93.62%
4	4033	3769	93.45%

Table 5: Whole dtrs. similarity, with unseen rules (WSJ-24)

The biggest factor is that 133 rules that occur in the heldout data have a score of 0, and 101 of these never appeared in the training data. Interestingly, however, this is out of 396 rules in WSJ-24 which have a frequency of 0 in the training data. In other words, although 0% of the unseen rules are predicted by frequency alone, 295, or 74.5%, of the new rules have a similarity score greater than zero. This indicates that similarity scores have potential in providing information for grammar generalization to new data. In this case, for example, a similarity score above 0 misses 133 rules that we need, but it picks up an additional 295. Future work can work on generalizing the grammar in such a way as not to overgeneralize.

Likewise, there are 634 rules in the BNC1000 which do not appear in the WSJ2-21 training data. A similarity score of 0 identifies 259 rules, of which 225 were not in the WSJ data. This means that 409, or 64.5%, of the unseen rules

have a score greater than 0. Again, we miss 259 rules, but gain 409 by looking at similarity scores above a threshold of 0.

4.3 Analyzing rare rules

To determine the effectiveness of the similarity scores on isolating structures which are not linguistically sound, as opposed to simply identifying ungeneralizable rules, we sample 100 WSJ rules occurring only once in the training data. Without examining the rule scores, we mark each as an error, ungrammatical, unclear, or correct. Of these 100, 21 are errors, and 5 covered ungrammatical constructions. For the bottom quarter of cases identified by the original methods, the whole daughters (similarity scores ≤ 24) method finds 7 errors, or 33% of them, additionally finding 2 ungrammatical cases. For the new method which does not use equivalence classes, we find essentially the same results: the bottom 22 cases (scores $= 0$) contain 8 errors, as well as 3 ungrammatical structures.

100 BNC rules which occur once in the WSJ training material were also sampled: 30 of these contain an annotation error, 46 are annotated correctly and the remaining 24 are unclear cases, often idiomatic phrases, inadequately covered by the Penn Treebank guidelines. As with the WSJ data, the similarity scores are reasonably well aligned with the erroneous cases: the bottom third of cases with the original method (similarity scores < 3) contain 13 of the 30 erroneously annotated rules. The bottom third of cases (similarity score=0) with the new method contain 14 of these 30 rules.

We can thus see that similarity-based scores are, in practice, effective at sorting problematic low-frequency rules from less problematic ones. Taken together with the results from the previous section that similarity sorts low-frequency rules in terms of ungeneralizability, we can see that similarity is a useful feature for determining a treebank grammar.

Where similarity does not work But what about the errors that have high similarity scores? It turns out that these types of rules are those which happened to be errors, but which actually license legitimate structures. For example, the structure in (4) seems to be erroneous because *just* modifies the NP. However, S $\rightarrow$ ADVP NP PP (score of 154) is a potentially legitimate structure: S $\rightarrow$ NP PP is licensed in the case of sentence gapping and small clauses (Bies et al, p. 127, 252ff), and a sentential adverb (ADVP) would attach as a sister of the NP and PP (Bies et al, p. 13).

(4) And the company is certain * to get out some aircraft with [$_S$ [$_{ADVP}$ just] [$_{NP}$ supervisors and other non-striking employees] [$_{PP}$ on hand]] .

Equivalences classes With this test-bed of cases, we can also analyze how the method changes by removing the equivalence class criteria, in both positive and negative ways. Starting with rules which are errors, we can see how they are more appropriately receiving lower scores. Consider example (5), with the rule NP → JJS JJ NN VBZ. The POS category VBZ is clearly wrong, but the original whole daughters method assigns it the similarity score 1,547, because the reduced rule NP → JJ NN VB is similar to lots of NP → JJ NN rules. By keeping the rule distinct, the new whole daughters score is 7, because this exact sequence is difficult to match.

(5) [$_{NP}$ most/JJS Western/JJ air/NN fleets/VBZ]

Consider also (6), with the rule NP → NP -LRB- NP , NP -RRB-. The whole daughters score goes from 10,462 to 0. The equivalence mappings would reduce this rule to NP → NP NP NP, thus losing crucial information about how bracketing should be done for parenthetical information. Although parentheses are "always adjuncts," they are informative adjuncts.

(6) [$_{NP}$ [$_{NP}$ Rep. Ronnie Flippo] -LRB-/-LRB- [$_{NP}$ D.] ,/, [$_{NP}$ Ala.] -RRB-/-RRB-] , one of the members of the delegation , says ...

>From the BNC data, consider cases like (7a), which contains the erroneous rule NP -> " NN " (with the corrected version in (7b)). With the old method, this is reduced to the extremely frequent NP -> NN, resulting in a similarity score of 10,802. Without equivalence classes, it receives a score of 8. The new method proves useful in identifying the incorrect annotation of quotations.

(7) a. [$_{NP}$ [$_{NP}$ "/" wardrobe-woman/NN "/"] [$_{PP}$ at the school]]
b. [$_{NP}$ "/" [$_{NP}$ wardrobe-woman/NN] "/" [$_{PP}$ at the school]]

There are cases, on the other hand, in which the new method performs less well, assigning low scores to correct rules. For instance, in example (8) from WSJ-24, with the correct rule S → -LRB- " NP " VP . -RRB-, the similarity score moves from 159,444 to 0. We see such a dramatic difference because of punctuation. The reduced rule was S → NP VP, which is clearly correct and similar to other rules.

(8) [$_{S}$ -LRB-/-LRB- "/" [$_{NP}$ Quest for Fire] "/" [$_{VP}$ was the first time] ./. -RRB-/-RRB-]

While removing some punctuation may assist in comparing rules, it is not clear-cut, as the punctuation that is meaningful can differ across treebanks. And as cases like (6) and (7a) show, punctuation can be informative.

5 Summary and Outlook

We have furthered the work of ad hoc rule detection by making it more corpus-independent. We have also verified that similarity is a crucial factor in determining the reliability of a rule, providing information unavailable in frequency, including a way to score rules which are not in the training data.

Given the success of such a method, a next step is to verify its effectiveness on other treebanks. An additional question is to see what effect these scores have on parser training, either by filtering rules identified by such methods, or using them in a parse reranking model . Since this work points to ways to generalize a grammar to new data, one can also explore the effects of the scores on parsing across genres (Sekine, 1997; Gildea, 2001; McClosky et al., 2006) and their applicability to active learning techniques (e.g., Tang et al., 2002).

Acknowledgements

Thank you to the three anonymous reviews for their helpful comments.

References

Bender, Emily M., Dan Flickinger, Stephan Oepen and Timothy Baldwin (2004). Arboretum: Using a Precision Grammar for Grammar Checking in CALL. In *Proceedings of the InSIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*. Venice, Italy.

Bies, Ann, Mark Ferguson, Karen Katz and Robert MacIntyre (1995). *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.

Burnard, Lou (2000). *User Reference Guide for the British National Corpus*. Tech. rep., Oxford University Computing Services.

Daelemans, Walter, Antal van den Bosch and Jakub Zavrel (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning* 34, 11–41.

Dickinson, Markus (2008). Ad Hoc Treebank Structures. In *Proceedings of ACL-08*. Columbus, OH.

Dickinson, Markus and W. Detmar Meurers (2005). Prune Diseased Branches to Get Healthy Trees! How to Find Erroneous Local Trees in a Treebank and Why It Matters. In *Proceedings of TLT 2005*. Barcelona, Spain.

Foster, Jennifer and Josef van Genabith (2008). Parser Evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics. In *Proceedings of LREC 2008*. Marrakech, Morocco.

Foth, Kilian and Wolfgang Menzel (2006). Robust Parsing: More with Less. In *Proceedings of ROMAND 2006*.

Gildea, Daniel (2001). Corpus Variation and Parser Performance. In *Proceedings of EMNLP-01*. Pittsburgh, PA.

Hogan, Deirdre (2007). Coordinate Noun Phrase Disambiguation in a Generative Parsing Model. In *Proceedings of ACL-07*. Prague, pp. 680–687.

Marcus, M., Beatrice Santorini and M. A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330.

McClosky, David, Eugene Charniak and Mark Johnson (2006). Reranking and Self-Training for Parser Adaptation. In *Proceedings of COLING-ACL-06*. Sydney, pp. 337–344.

Sekine, Satoshi (1997). The Domain Dependence of Parsing. In *Proceedings of ANLP-96*. Washington, DC.

Tang, Min, Xiaoqiang Luo and Salim Roukos (2002). Active Learning for Statistical Natural Language Parsing. In *Proceedings of ACL-02*. Philadelphia, pp. 120–127.

Towards a Multi-Representational Treebank

Fei Xia
University of Washington
Seattle, WA 98195, USA
fxia@u.washington.edu

Owen Rambow
Columbia University
New York, NY 10115, USA
rawbow@ccls.columbia.edu

Rajesh Bhatt
University of Massachusetts
Amherst, MA 01003, USA
bhatt@linguist.umass.edu

Martha Palmer
University of Colorado
Boulder, Colorado 80309, USA
Martha.Palmer@colorado.edu

Dipti Misra Sharma
International Institute of Information Technology
Gachibowli, Hyderabad 500019, India
dipti@iiit.ac.in

Abstract

Computational, descriptive, and theoretical linguistics use both phrase (PS) structure and dependency structure (DS) to represent syntax. We believe that the next-generation treebank should be multi-representational, designed for both representations with an automatic conversion. In this paper, we highlight the assumptions made by existing PS-to-DS and DS-to-PS conversion algorithms and show the limitations of these algorithms. We then propose a new DS-to-PS conversion algorithm that outperforms existing algorithms and allows more flexibility. Our experiments and error analysis show that high-quality DS-to-PS conversion is possible if the conversion process is performed at the designing stage of treebank construction to ensure that all information we wish to represent in PS is provided in DS.

1 Introduction

Treebanks have profoundly changed and advanced the field of NLP, and parsing in particular. While most studies on statistical parsing in the 1990s used phrase-structure (PS) treebanks, recently there has been an explosion in interest in dependency structure (DS) parsing (e.g., [1, 5, 6, 2]) which require dependency treebanks. Therefore, we claim that the next-generation treebank should be *multi-*

representational; that is, the treebank should document clearly *what* information represented in the the treebank, and *how* it is represented in PS and in DS. If PS and DS represent the same information, then automatic conversion between them should be possible.

We are currently building a multi-representational treebank for Hindi/Urdu with the following strategy: at the guideline designing stage, we develop coordinated DS and PS annotation guidelines. A DS-to-PS conversion algorithm is developed and tested on a test suite and the examples used in the guidelines to ensure that PS and DS guidelines are consistent and that sufficient information (e.g., a rich set of dependency types) is included in the input DS to allow high-quality conversion. At the annotation stage, we will manually create DSs for the sentences, and the corresponding PSs will be generated automatically from the DSs by applying the same DS-to-PS conversion algorithm. We choose the DS-to-PS direction, rather than the other way around, because we believe that annotating DS is faster than annotating PS, and because of we can build on existing work [7]. The success of the strategy depends on the answers to three crucial questions:

(Q1) What does consistency between DS and PS mean? Are DS and PS always consistent?

(Q2) Is it possible to achieve high accuracy for DS-to-PS conversion?

(Q3) What kinds of information should be included in the DS to help with the conversions?

In this paper, we address these questions by first discussing the relation between PS and DS and defining consistency between the two, and then proposing a new DS-to-PS algorithm that outperforms existing algorithms and allows more flexibility. Finally, through an error analysis on the experiments, we show that high-quality DS-to-PS conversion is possible if the conversion process is performed at the designing stage of treebank construction to ensure sufficient information is provided in the input DS.

2 Relation between DS and PS

A *syntactic theory* (in the large sense) is an account of how to represent all sentences of a language in the chosen representational framework (PS or DS); there are many possible syntactic theories for each language, for both PS or DS. Some may be better than others, but there is no single PS or DS for any sentence, in any language. Figure 1 shows possible phrase and dependency structures for an example sentence: the former uses as its theory the English Penn Treebank guidelines and the latter is based on an extended version of various deep-syntactic representations.

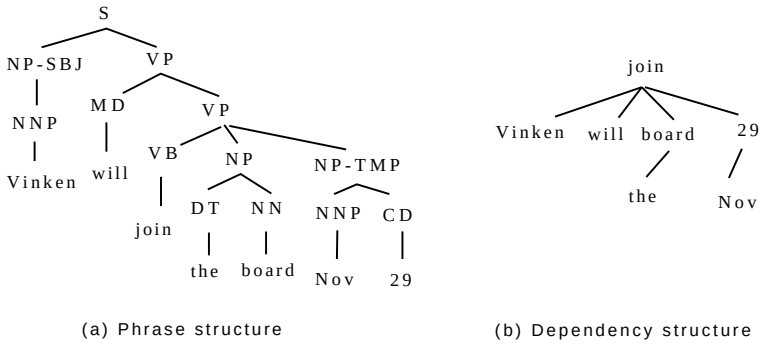


Figure 1: PS and DS for the sentence *Vinken will join the board Nov 29*

The two representations are clearly different. Formally, both PS and DS are types of trees: a DS is a tree whose nodes are all labeled with the symbols appearing in the sentence;[1] a PS is a tree whose leaf nodes are labeled with the symbols in the sentence and whose internal nodes are labeled with syntactic categories (e.g., *NP* four noun phrase). Linguistically, a PS groups consecutive words hierarchically into phrases (or constituents). In a DS, dependency between a head and its dependents is the primary relation represented, and the notion of constituent is only derived. Therefore, PS can represent projections (e.g., a noun phrase as a projection of nouns) directly as edges in the tree, but DS cannot. Similarly, DS can represent dependency relation directly as edges in the tree, but PS cannot.

Other differences that one often finds between PS and DS are *preferential*, not *definitional*. Both representations can have certain properties often believed to be characteristic of one type or the other. For instance, both can use or not use empty categories; both can add labels to edges; both can allow or disallow crossing edges; both (as trees) can be ordered or unordered.

A common PS-to-DS conversion algorithm, used by many parsing algorithms (e.g., [3]) to find head in a PS, works as follows: first, for each internal node in the PS, one of its children is chosen as the head child; next, starting from the leaf nodes of the PS, the head word is propagated from the head child to its parent; finally, a DS is created by making the head word of each non-head-child depend on the head word of the head-child. Given the PS in Figure 1(a), Figure 2(a) shows the tree after the second step, and the resulting DS is in Figure 1(b). This algorithm assumes that the input PS, once *flattened* and with syntactic labels removed, is identical to the desired DS. By *flatten*, we mean that all the internal nodes in the PS are merged

[1]For the sake of this definition, we assume that the surface sentence can (but need not) contain empty categories (i.e., symbols with null phonological value such as trace).

with their head child. In this example, after flattening the PS in Figure 2(a), the new PS, as shown in Figure 2(b), is identical to the DS in Figure 1(b) except for the syntactic labels.

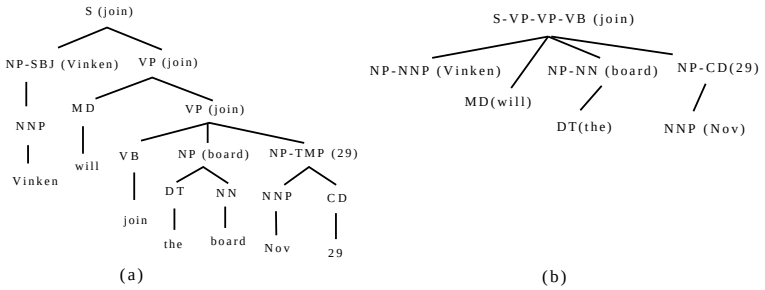


Figure 2: A PS after head word is propagated from bottom up and the flattened version of the PS

A PS can correspond to several flattened trees depending on what node is chosen as head child. We call a PS and a DS *consistent* if there exists a flattened tree for the PS that is identical to the DS once syntactic labels are removed. Given a sentence, if the desired PS and the desired DS are consistent, we say that the *consistency assumption* holds for the sentence. The question is whether consistency assumption always holds. The answer would depends on linguistic choices made in designing the PS and DS. Taking long-distance wh-movement as an example, Figure 3(a) is the PS according to the English Penn Treebank [4]. It uses coindexation with an empty category to represent the long-distance dependency. In contrast, (b) is a DS that uses non-projectivity to represent the long-distance dependency. It is easy to show that the DS and the PS in (a) are inconsistent.[2]

Some of the inconsistency between PS and DS can be handled by creating a new DS that is consistent with the PS. For instance, we can make the DS in 3(b) consistent with the PS in (a) by using the same device to represent the long-distance depend as the DS, namely coindexation with an empty category, as shown in 3(c). (We could also have used non-projectivity in the PS.) In our Hindi/Urdu treebank project, we plan to test whether *all* kinds of inconsistency between PS and DS can be handled by creating a new DS′ which is consistent with the desired PS and whether the process can be totally automated. We expect this to be the case if all information that we want to represent in PS is also represented in DS or can be derived from the syntactic theory underlying the PS.

[2]In order for *who* to depend on *come* as in Figure 3(b), *come* has to be head of the whole sentence, which contradicts with the fact that *think* is the root node in the DS.

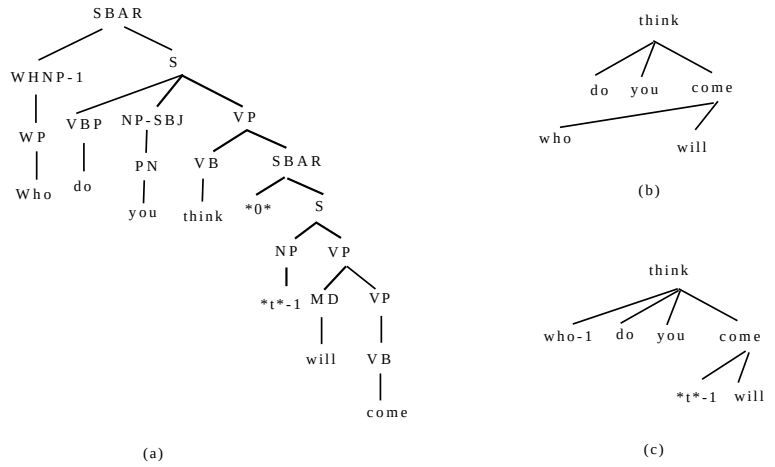


Figure 3: The PS and two DSs for *Who do you think will come*

3 A new DS-to-PS algorithm

Xia and Palmer [8] compared three DS-to-PS algorithms, which all could be seen as the reverse of the *flattening* process in the PS-to-DS algorithm: the algorithms expanded each node in the input DS into a projection chain, and labeled the newly inserted nodes with syntactic categories. They differ in the heuristics adopted to build projection chains and attach the PS subtree for a dependent to the subtree for the head.

These algorithms have two major limitations. First, they all make additional assumptions besides the consistency assumption; for instance, they all assume that each POS tag has exactly one possible projection chain. The assumptions turn out to be too strong and the exceptions to the assumptions are the main sources of errors [8]. Second, the input DS may use a rich set of dependency types, but the algorithms cannot take advantage of the information because they distinguish only arguments and modifiers. We propose a new algorithm that overcomes these limitations. (Like the previous algorithm, we assume the DS is consistent with the target PS.) Compared to existing algorithms, the new algorithm only assumes that the consistency assumption holds and any exception to the assumption is handled by a preprocessing step that generates a new DS that is consistent to the desired PS (see Section 2). The consistency assumption states that the input DS is identical to a flattened version of the desired PS; therefore, there is a mapping between the segments in the DS and the segments in the PS. Based on this assumption, the new algorithm works by decomposing the input DS into multiple DS segments,

replacing each DS segment with the PS counterparts, and *glue* these PS segments together to form a PS.

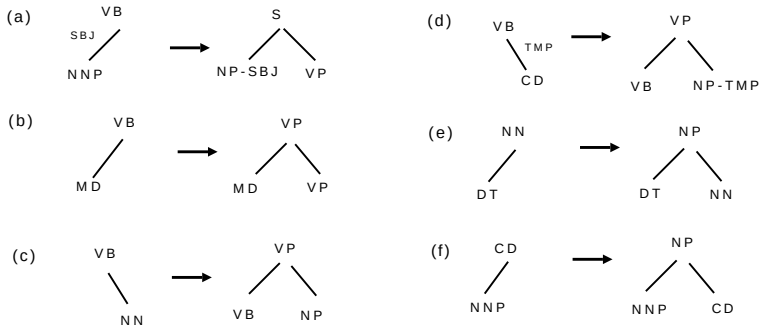


Figure 4: Conversion rules derived from the PS in Figure 2(a)

3.1 Conversion rules

A conversion rule is a (DS pattern, PS pattern) pair. In our current experiments, the DS pattern is simply a dependency link with associated information such as dependency type and the POS tags of the head and the dependent, and a PS pattern is one-level tree with a root node and two children. In Figure 4(a), the DS pattern means that an NNP depends on a VB and the dependency type is SBJ. The corresponding PS pattern includes a parent node *S* and two of its children *NP* and *VP*; the function tag of the *NP* comes from the dependency type in the DS pattern.

Conversion rules can be created by hand, or extracted from existing PS. The rule extraction algorithm is the same as the PS-to-DS algorithm in Section 2 except that the last step is replaced with the following, as illustrated in Figure 5: for each internal node ZP in the PS with more than one child, let XP be ZP's head child and X be the POS tag of XP's head word. For every non-head child YP of ZP, let Y be the POS tag of YP's head word, the conversion rule in Figure 5(b) is created. Figure 4 shows the rules extracted from the PS in Figure 2(a).

3.2 Applying conversion rules to form PS

Given a DS, the new conversion algorithm selects a conversion rule for each dependency link, and combines the PS patterns of the rules to form the output PS. The algorithm is as follows:

Input: An input DS and a set of conversion rules
Output: An output PS

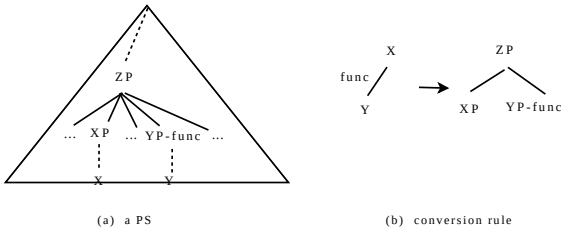


Figure 5: Extracting conversion rules from a PS

Algorithm:
(1) if X, the root of the DS, is a leaf node
(2) the output PS contains only the node X; return
(3) for each child Y of X
(4) build a PS T_Y for the DS subtree rooted at Y
(5) initialize a projection chain, proj_chain, that contains only X
(6) for each left child Y of X in the DS, starting from right to left
(7) (a) choose a conversion rule r for (Y,X) based on the projection chain
(8) (b) apply the rule by updating the projection chain and attaching T_Y
(9) Same as (6)-(8), but it is for each right child Y of X, going from left to right
(10) Consolidate the dominant links in the projection chain

Given the DS in Figure 6(a) and the conversion rules in Figure 4, Line (3)-(4) builds the PS trees for the dependents of *join*, as shown in Figure 6(b1)-(b4). Line (5) builds the initial projection chain as in Figure 7(a), and each iteration of Line (6)-(8) (and (9) for the right children) will attach one subtree in Figure 6 to the chain, and update the chain accordingly, as shown in 7 (b) through (e). The circle marks the lowest position on the projection chain that allows future attachment of subtrees for dependents.

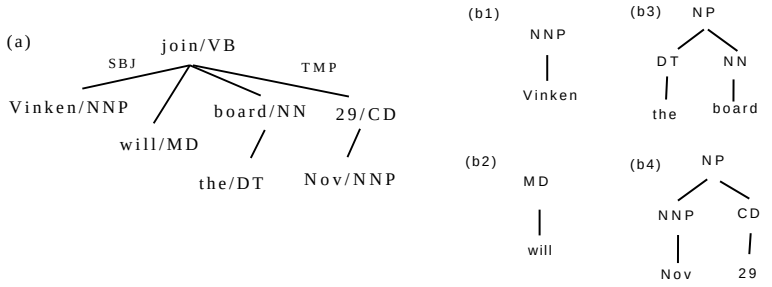


Figure 6: The input DS for the new algorithm and the subtrees built by Line (3)-(4) of the new algorithm

4 Experimental results

Ideally, to test a DS-to-PS algorithm, we shall use a treebank with both DS and PS representations. However, since such a treebank does not exist, we followed the same practice as in [8]: we first convert the PS in the English Penn Treebank (PTB) to DS with the PS-to-DS algorithm in Section 2 where the head child is chosen according to a head percolation table [3], then build a new PS from the DS with the new DS-to-PS algorithm, and finally compare the new PS with the original PS.

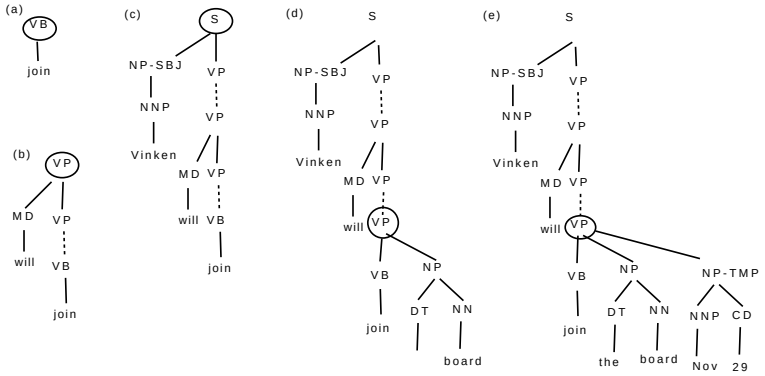


Figure 7: Results after attaching the dependents of *join*

4.1 Creating and selecting conversion rules

If multiple rules in a conversion rule set share the same DS pattern, we call the DS pattern *ambiguous*. The ambiguity can be reduced by including more information in the DS pattern. For instance, a DS pattern can include

(P1) the POS tags of the dependent and the head, and the position (left or right) of the dependent with respect to the head.

(P2) all the information in (P1) plus the dependency type (e.g., -SBJ).

(P3) all the information in (P2) plus a flag indicating whether the dependent is a leaf node in the DS.

(P4) all the information in (P3) plus a flag indicating whether there are other dependents between this dependent and the head in the DS.

For dependency links in the input DS that match multiple rules, one could build a sophisticated model for rule selection; however, because we believe that most, if

Table 1: Conversion results on Section 22 (*Labeled* precision/recall/fscore)

	# of rules	# of patts	Non-cheating		Cheating	
			(S1)	(S2)	(C1)	(C2)
(P1)	1841	965	72.9/90.5/80.7	81.9/86.4/84.1	85.5/87.6/86.5	100/98.2/99.1
(P2)	2507	1645	75.0/92.6/82.9	83.6/88.9/86.2	88.1/90.7/89.4	100/98.2/99.1
(P3)	2826	1996	79.7/92.4/85.6	86.9/90.5/88.7	91.2/92.5/91.8	100/98.2/99.1
(P4)	3513	2626	80.8/92.8/86.4	88.1/90.7/89.4	92.9/92.9/92.9	100/98.2/99.1

not all, of the ambiguity could be eliminated for our treebank project (c.f. Section 4.2), we adopted two simple strategies as follows:[3]

(S1) Always choose the most frequent rule for the DS pattern assuming the frequency of rules is available from the training data.

(S2) Choose the rule based on the current projection chain that is under construction. The algorithm prefers rules that add fewer number of nodes and attach the subtree lower.

4.2 Results and Error Analysis

We extracted conversion rules automatically from the phrase structures in one section, Section 19, of the PTB, and ran the new conversion algorithm on another section, Section 22, of the PTB.[4] The results are in Table 1. The experiment shows that when more information is included in the DS pattern, the number of rules increase, as does the performance of the algorithm. Also, (S2) outperforms (S1) for all the rule sets.

As [8] reported the results of Algorithms 1-3 on Section 00, we ran our algorithm on the same section with the (P4)+(S2) setting (the rules were extracted from Section 19). The results are shown in Table 2. Here, we use *unlabeled* precision/recall/fscore following [8]. The experiments show that the new algorithm outperforms all the previous algorithms.

The errors made by our system come from three sources:

(R1) Missing rules: Some of the conversion rules needed to produce the gold standard PS for the test data are not in the rule set.

(R2) Wrong rules selected: The correct conversion rules are in the rule set, but are not selected by the system.

[3]For dependency links in the input DS that do not match any rules, the algorithm forms a conversion rule on the fly to ensure that a PS is produced.

[4]We did not use a larger data set for rule extraction because using more data or different data does not change the results substantially.

Table 2: Conversion results on Section 0 (*Unlabeled* precision/recall/fscore)

	Precision	Recall	Fscore
Alg1	32.81	81.34	46.76
Alg2	91.50	54.24	68.11
Alg3	88.72	86.24	87.46
New alg	89.19	91.76	90.46

(R3) Exception to the preferences: For ambiguous DS patterns, the algorithm prefers rules that add fewer number of nodes and attach subtrees lower. Gold-standard PS might have different preferences.

To tease apart the effect of each type of errors, we ran two cheating experiments: in (C1), the *missing* rules are added to the rule set; in (C2), for each dependency link in the input DS, the correct conversion rule for the link is provided as part of the input. The results with (S2) for rule selection are shown in the last two columns of Table 1. In this table, the gap between (C1) and (S2) is caused by (R1), the gap between (C1) and (C2) is caused by (R2), and the gap between (C2) and 100% is caused by (R3). The experiments show that (R2) is the main source of errors, and the f-score in (C2) is very close to 100%, implying that the heuristics adopted by the algorithm to glue PS segments together work very well.

Because most errors are caused by (R2), it is important to understand what causes ambiguity in DS patterns and whether the ambiguity can be eliminated. To answer the question, we compare the rules selected by the gold standard with the rules selected by the best non-cheating system (i.e., (P4)+(S2)).Out of the 42,675 dependency links in Section 22, the system chooses wrong rules for 3,407 links, resulting in an error rate of 7.98% in selecting rules. These correspond to 1,043 link types. We manually checked the most frequent 100 link types (which account for 53.07% of the link tokens) and classified them into four categories.

Missing content (13.2% of error tokens): If the PS makes a syntactic distinction which the DS does not make, then the conversion from DS to PS cannot be correct. For example, if the PS distinguishes an SBAR relative clause from an SBAR complement clause by attaching them at different projections of a noun, and the DS does not make this distinction (for example, using arc labels), then there can be no automatic conversion from that DS to that PS.

Coordination (14.7% of the error tokens): It is a well known fact that it is not easy to distinguish scope in coordination structurally in a DS (for example, the distinction between "*young (men and women)*" and "*(young men) and women*)". This is another case of missing information: if syntactic structure is not annotated at DS, it cannot be automatically created at PS.

Inconsistency in the PTB (7.9% of the error tokens): For instance, when an

adverb appears between the subject NP and the verb V, the adverb may or may not project to an *AdvP*, and the *AdvP* may attach to a *VP* or a *S*. As a result, a DS pattern may correspond to several PS patterns. This type of ambiguity is a by-product of the current experiment setup: conversion rules are extracted from a PS treebank that contain this kind of inconsistency, and automatic creation of the PS from DS will in fact eliminate such inconsistencies.

Punctuation (17.4% of error tokens): When the dependent is a punctuation mark, the DS pattern is often very ambiguous because punctuation marks can appear in many places and it is hard to provide an informative dependency type between them and their heads, a problem that we plan to study in the future.

5 Conclusion and future work

Because of the usefulness of PS and DS, we believe that the next-generation treebank should be multi-representational. We propose to build such a treebank by creating PS and DS guidelines simultaneously (along with rules that convert DS to PS), building DS manually and then converting DS to PS automatically. In this paper, we address several questions that are crucial for the success of this strategy. First, we define the consistency between PS and DS, and show that the consistency assumption, adopted by all existing DS-to-PS and PS-to-DS conversion algorithms, does not always hold. Second, we propose a new DS-to-PS conversion algorithm that is more flexible and outperforms previous algorithms; furthermore, conversion rules used by the algorithm can be extracted automatically from PS. Third, the error analysis shows that most, if not all, of ambiguity in DS patterns could be eliminated if DS and PS guidelines are designed at the same time (to ensure that we represent the same information at DS and PS). We can use the DS-to-PS conversion process during the guideline design time to detect potential missing information in the input DS and inconsistency between PS and DS.

For the future work, we plan to implement a preprocessor to transform the DS into a PS-consistent DS'; and to test and improve the new algorithm while working on the Hindi/Urdu treebank project. For instance, the tree examples in the PS and DS guidelines would serve as the initial test set for our conversion algorithm. We will run the rule extraction algorithm to detect any ambiguous DS patterns and determine whether the ambiguity could be totally eliminated. We will also study whether the DS pattern needs to be extended beyond a single dependency link.

References

[1] J. Hajič and E. Hajicova and M. Holub and P. Pajas and P. Sgall and B. Vidova-

Hladka and V. Reznickova. The Current Status of the Prague Dependency Treebank. In *Lecture Notes in Artificial Intelligence (LNAI)*, volume 2166, pages 11–20. Berlin, Heidelberg, New York, 2001.

[2] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June 2008.

[3] David M. Magerman. Statistical Decision-Tree Models for Parsing. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, Cambridge, Massachusetts, USA, 1995.

[4] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[5] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT/EMNLP*, 2005.

[6] J. Nivre, J. Hall, and J. Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *In Proceedings of LREC*, pages 2216–2219, 2006.

[7] Dipti Misra Sharma, Rajeev Sangal, Lakshmi Bai, and Rafiya Begam. Anncorra: Treebanks for indian languages (version - 1.9). Technical report, Language Technologies Research Center IIIT, Hyderabad, India, 2006.

[8] Fei Xia and Martha Palmer. Converting Dependency Structures to Phrase Structures. In *In the Proc. of the Human Language Technology Conference (HLT-2001)*, San Diego, CA, 2001.

To Use a Treebank or Not – Which Is Better for Hypernym Extraction?

Erik Tjong Kim Sang
University of Groningen
The Netherlands
E-mail: e.f.tjong.kim.sang@rug.nl

Abstract

We compare two processing methods for a single natural language processing task. One uses a treebank created with a full parser while the other restricts itself to lexical and part-of-speech information. We show that for the task under investigation, automatic extraction of hypernym-hyponym pairs from text, the former does not outperform the latter. We compare the output of the two approaches and look for an explanation for this unexpected result.

1 Introduction

In recent years there has been an increased availability of treebanks created by advanced parsing tools. These offer detailed syntactic relations between words and phrases, unlike text corpora of about fifteen years ago which only offered words and their syntactic classes. We expect material from treebanks to be very useful for various natural language processing tasks, more than for example text that has just been processed by a part-of-speech tagger. In order to validate our expectation, we set up an experiment in which we compared the performance of the two in a single task. Contrary to our expectation, the output obtained from the treebank material turned out to be less reliable than that of the tagged text.

This paper describes the comparison experiment. After this introduction, we introduce the target task, extraction of hypernymy information from text, and describe how the experiment was set up. Next, we present the results of the comparison and discuss possible reasons for the outcome. In the final section, we present some concluding remarks.

2 Methods and experiments

We examine a single task: extraction of hypernymy information from text. A hypernym of a word X is a word Y which both contains the meaning of X and is broader. For example, an *orange* is a *fruit*, thus the word *fruit* is a hypernym of the word *orange*. If Y is a hypernym of X then X is a hyponym of Y. So *orange* is a hyponym of *fruit*.

Information about related hypernyms and hyponyms can be found in lexical resources such as WordNet [1] and EuroWordNet [8]. However, these resources are incomplete and therefore it is interesting to look for additional hypernymy information. One method for obtaining such information is to look in text for context patterns that link related words [3]. A phrase like *Y such as X* often contains a good hypernym candidate for *X*, namely Y.

For our work, we have used an extraction technique proposed by Snow, Jurafsky and Ng [6][1]. They searched in a large text corpus for sentences containing related word pairs, recorded the contexts of these pairs and used the context information for finding new pairs of related words. For example, if the words *orange* and *fruit* are known to be related and if the corpus contains the phrase *oranges and other fruits* then the phrase *X and other Y* can be used for finding candidate hypernyms for the word *X*.

Collecting phrases from text is a matter of counting how often a phrase appears in a text, either with related words or with unrelated words. Lexical resources usually do not contain information about unrelated words. Like Snow et al., we assume that two words are unrelated if they are both present in the resource while they have not been defined as related [6]. Therefore we only consider evidence for context phrases that relate two known words.

We combine the evidence of different context patterns in order to determine if two words are related. For this purpose, we apply the machine learning method Bayesian Logistic Regression [2] (also used by [6]). However, any other machine learner can be used for this task. Evaluation is performed with 10-fold cross validation which means that we divide the available material in ten sections and use each section as test set with the other nine as training set. The performance is measured by counting how many hypernym-hyponym pairs are found and how many of them are correct. We combine these two counts in precision scores. Since we obtained various different test set sizes, we also registered the number of positive cases (targets) of each test set.

The main goal of the experiment was to compare information available in a treebank that was built with a full parser, with linguistic annotations obtained from

[1] Similar work has been done by Nichols et al. [4].

a part-of-speech tagger. We used Dutch NRC newspaper section from the Alpino Treebank which was created by the parser with the same name [7][2]. The treebank contains 5.7 million sentences with over 100 million tokens. Like in the work of Snow et al., we limited the maximum size of the context phrases to four dependency links. Additionally, a single word modifying one of the two target words could be added to the phrase. An example of a context phrase is *Y like(modifies Y) X(complements like)*. Rather than the words in the text, the lemmas of these words were used in the phrases. We only used head words of noun phrases as target words.

The second data source was obtained by processing the same newspaper corpus with a part-of-speech tagger and lemmatizer that were trained on the Dutch CGN corpus as described in our earlier work [5]. A basic filter was used for identifying noun phrases: Det* Adj* N+. Like with the treebank data, target phrases consisted of lemmas, and linked head words (the final word) of noun phrases. These phrases were restricted to three center words to which a single word that modified one of the target words could be added. An example of such a phrase is *Y like X the(modifies X)*. We considered all nonhead words of the noun phrases as possible modifiers.

3 Results and discussion

We ran two hypernym extraction processes on the Dutch newspaper corpus, one using the treebank material while the other used the output of the part-of-speech tagger. Context patterns were extracted using hypernym-hyponym pairs taken from the Dutch part of EuroWordNet [8]. These patterns were used to extract candidate hypernym-hyponym pairs from the corpus. The evidence for these pairs was combined with Bayesian Logistic Regression and the results were evaluated using ten-fold cross-validation.

The results of the experiments can be found in Table 1. Since the extraction process produced different numbers of suggestions for the two data sources, we have performed additional evaluations where the number of proposed hypernym candidates was increased or decreased to match the size corresponding with the other data set (by changing the acceptance threshold of the machine learner). We registered a significantly higher precision score for the tagged data (41.8–18.6). However, the differences were a lot smaller when we corrected the numbers for the data size. We only registered a significant difference between the precision scores for the tagged data in comparison with the reduced output of the treebank data (41.8–33.3, $p < 0.05$).

[2]The output of the parser has *not* been manually corrected. The accuracy of the parser is about 90% F score for labaled dependency relations.

Data source	Targets	Found	Correct	Precision
Tagged data	675±24	225±15	94±9	41.8±3.7%
(adjusted threshold)		905±29	183±14	20.2±1.3%
Treebank data	1027±32	905±28	168±13	18.6±1.3%
(adjusted threshold)		225±15	75±9	33.3±3.3%

Table 1: Hypernym extraction results for the NRC newspaper corpus: number of related word pairs in the test set, number of extracted pairs, number of correct pairs, precision score and the associated standard deviations. There are two result lines for each data source: one with the default acceptance threshold and one with an adapted threshold to obtain the same number of accepted word pairs as with the other data source.

The extraction process that used the treebank data failed to outperform the process that only had access to tagged data and even failed to reach the same precision levels. We were surprised about this fact. How could the treebank patterns be more inaccurate than the lexical patterns? In order to find an answer to this question, we examined the hypernym-hyponym pairs suggested by the two best individual patterns measured by $F_{\beta=1}$ rate. Both the treebank data and the tagging data had the pattern *X and other Y* as best individual pattern. The precision scores of the two patterns were similar (23–25) but the treebank pattern missed more pairs found by the lexical pattern than vice versa (38–34) and generated more additional incorrect pairs (142–120, see Table 2).

From the output of the two approaches, we examined the correct pairs that were proposed by the lexical pattern but were missed by the treebank pattern. These are the pairs which potentially could have caused the precision of the treebank pattern to be lower than it should have been. There were 38 of such pairs. After examining them, we found four different causes for the misclassifications. Three of the pairs were missed because of two different causes.

Three of the four problems had their origin in processes outside of the parser. Sixteen pairs (42%) were indentified by the treebank pattern but were omitted from the test data because there were insufficient other patterns to support them (the process requires a minimum of five different extraction patterns to support a new hypernym-hyponym pair). One pair (3%) was missed because of a parse tree feature that was missing from the extraction software. And eight pairs (21%) were missed because of lemmatizing problems.

Sixteen (42%) of the missed hypernym-hyponym pairs originated from parsing problems. A frequent structure in the text is *X of Y and other Z*. The lexical pattern would always combine *Y* with *Z* but the treebank pattern could propose the pair *X*–*Z*

	correct		incorrect	
	+tree	-tree	+tree	-tree
+lexical	52	38	143	120
-lexical	34	–	142	–

Table 2: Confusion matrix for the best-performing lexical pattern *X and other Y* and the corresponding treebank pattern (tree). The matrix shows how many word pairs were found (+) or missed (-) by each of the two types of extraction patterns and, additionally, how many of these word pairs were related (correct) and how many were not.

as well as *Y–Z*, depending on the underlying tree structure. Any of these two could be correct but the examples from our data seem to suggest that shorter distance relations (like *Y–Z*) are more likely than longer distance relations (*X–Z*). Any time when the treebank patterns suggest a longer distance relation, they are predicting a low probability event. This could be an explanation for the fact that the treebank patterns achieve lower precision scores than lexical patterns which only suggest the more likely shorter distance relations.

We will forward the treebank pattern problems to the authors of the treebank. Some of these are solvable, like the attachment of *such as* to numbers like in *a million animals such as geese*. However, others are caused by tasks which are hard for any language processing system, like for example prepositional phrase attachment. We do not expect all parsing problems to be solved soon.

4 Concluding remarks

We have compared the effects of data sources on a single linguistic task: the extraction of hypernymy information from text. We compared extraction patterns built from a treebank with patterns that were generated from part-of-speech tagger output. When applied to a Dutch newspaper corpus, we found that, contrary to our expectations, the treebank patterns did not outperform the lexical patterns and failed to reach the same precision scores. Inspection of the data suggested that cases missed by the treebank patterns were caused by parsing errors, some of which will be hard to avoid.

We believe that treebanks are useful resources for natural language processing tasks and that they should enable higher quality output than data annotated with more shallow tools than a full parser. We hope that this study will contribute to improving treebank quality and to the knowledge of how they can be applied for

supporting natural language processing tasks. In the future, we will remain working towards these goals.

References

[1] Christiane Fellbaum. *WordNet – An Electronic Lexical Database*. The MIT Press, 1998.

[2] Alexander Genkin, David D. Lewis, and David Madigan. *BBR: Bayesian Logistic Regression Software*. www.stat.rutgers.edu/˜ madigan/BBR/, 2005.

[3] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of ACL-92*. Newark, Delaware, USA, 1992.

[4] Eric Nichols, Francis Bond, Takaaki Tanaka, Sanae Fujita, and Daniel Flickinger. Robust ontology acquisition from multiple sources. In *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. Sydney, 2006.

[5] Erik F. Tjong Kim Sang. *Generating Subtitles from Linguistically Annotated Text*. Internal report Atranos project, WP4-12, University of Antwerp, 2003.

[6] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2005*. Vancouver, Canada, 2005.

[7] Gertjan van Noord. At last parsing is now operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, 2006.

[8] Piek Vossen. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publisher, 1998.

Semantic Annotation of Genitive Attributes in a German Treebank

Maya Bangerter
University of Zurich
Institute of Computational Linguistics
E-mail: bangerter@cl.uzh.ch

November 25, 2008

Abstract

German genitive attributes are usually tagged as such in treebanks. However, it is well known that this information is not sufficient for determining the type of relation between head nouns and attributes, as genitive attributes can express many different semantic relations. Various linguistic classifications have been worked out, but to my knowledge, nobody has so far proposed to apply this linguistic knowledge to a corpus. The challenge here is to come up with a classification that is both easy to verify and sufficiently fine-grained. Using earlier linguistic approaches as guidelines, I propose in this paper a detailed annotation scheme for German genitive attributes based on readily identifiable noun features. First insights from its application to the Smultron Treebank show that it is easy to distinguish between the proposed classes and that my classification of genitive attributes can be related to a more general semantic annotation level.

1 Introduction

In recent years a lot of work was done in the improvement of parsers using corpus linguistic resources. Today, semantic annotations are becoming increasingly important. More semantic information would be a great advantage for many NLP applications. The knowledge of implicit semantic relations is, for example, crucial for the quality of question-answering systems. There is no other way to match a potential answer to a question such as in the pair below. [1]

[1] Example sentence from Smultron, Literary Part.

(1) was bedeutet der schwarze Zylinder?
What does the top hat signify?

(2) ... der schwarze Zylinder des Universums ist...
... the top hat of the universe is...

However, the difficulties in relation mining start with determining a classification. In most cases, semantic relations are classified according to a specific domain. There is no consensus among linguists about a general classification of semantic relations. But ad-hoc classifications done in computational linguistics suffer from the lack of linguistic foundation.
In German, genitive attributes are often used to express implicit semantic relations. The genitives extend a core noun phrase. They occur pre- and postnominally, always adjacent to the head noun. Genitives are easily identifiable and therefore it is possible to nest them arbitrarily deep. Nouns can take two genitive attributes in German.
In German language studies, various attempts to classify these attributes have been undertaken. The results are quite different in detail, but the classifications share a large fraction of classes. These can already be very useful to NLP applications.
In the following, I will first discuss advantages and drawbacks of two linguistic classifications, the semantic typology of Helbig/Buscha and the syntactic one of Lindauer. Then I will show how one can arrive at partitioning of similar granularity using exclusively formal criteria and describe first experiences with the annotation of a German corpus. In the last section I will briefly outline how the new distinctions could be annotated automatically using machine learning.

2 Linguistic Approaches

Semantic classification For building their typology of genitive attributes, Helbig and Buscha [6] rely on their *predicative deep structure*, i.e., they paraphrase the attributes as predicates in order to turn implicit semantic relations into explicit ones. According to Helbig/Buscha there are twelve different possibilities; for example, the possessive genitive is based on a relation of ownership ("Haben-Verhältnis") and the defining genitive is based on an is-a relation ("Sein-Verhältnis").[2]

(3) das Haus meines Vaters ← mein Vater hat ein Haus
the house of my father ← my father has a house

(4) die Pflicht der Dankbarkeit ← Dankbarkeit ist eine Pflicht
the duty of gratefulness ← gratefulness is a duty

[2] An exhaustive presentation of all possible classes is given in [6, p. 591]

More specific classes are based on more specific predicates, e.g., the explicative genitive is traced back to a relation of signification ("Bedeuten-Verhältnis") and the 'genitivus auctoris' is based on a relation of creation ("Verhältnis des Schaffens"):

(5) der Strahl der Hoffnung ← der Strahl bedeutet Hoffnung
the ray of hope ← the ray signifies hope

(6) das Werk des Dichters ← der Dichter schuf das Werk
the opus of the poet ← the poet created the opus

There has been much debate on the value of such classifications; Eisenberg [3] , for example, criticizes it as "purely descriptive" and "without explicative value". Because the classification relies on paraphrasing, it is indeed hard to reproduce. This is its major drawback. Nevertheless, the typology of Helbig/Buscha provides a detailed and widely accepted linguistic classification scheme, when it comes to classifying semantic relations between nouns in a more general sense (as, e.g., in SemEval 07 [5]), even though it is constrained to a subclass of relations between nouns.

Syntactic classification Lindauer [7] develops a formal classification for genitive attributes in the context of generative grammar. His goal is a classification based only on morphologic and syntactic criteria. He distinguishes *thematic genitives* from all other genitives. Lindauer's analysis results in three simple syntactic tests, which can be used to positively determine possessive genitives and to determine partitive genitives *ex negativo*. The tests are as follows:

1. Possessive genitives can be replaced by a possessive pronoun.
2. Possessive genitives can be replaced by a prepositional phrase headed by the preposition 'von'.
3. Possessive genitives can occur in prenominal positions.

Lindauer points out that the dependencies between the nouns in partitive constructions are rather unclear. This is reflected in the wide variation between appositive and attributive forms of partitive attributes and verb agreement alternating between the head noun and the dependent noun. For these reasons, partitives certainly have to be considered separately in syntax.

The Duden Grammatik [3] follows Lindauer's new formal classification insofar as it retains a very broad concept of possessives. The group of partitives, however, is subdivided by semantic criteria.

[3] Eisenberg, cited by Lindauer [7, p. 138]

3 Annotation Scheme

A classification which serves as an annotation scheme has to be exhaustive, its classes have to be as selective as possible, it should agree with syntactic analyses and be semantically adequate. Lindauer's purely formal classification is a good starting point for such an approach but it is semantically too coarse and neither compatible with the most important syntactic analyses nor does it fit in a more general classification in the field of relation mining. We will therefore refine it using readily available noun features.

Deverbal nouns Genitive constructions involving deverbal nouns constitute a large fraction of genitive attributes. The genitives fill the argument slots of the deverbal noun which it inherits from the underlying verb. If there is only one attribute attached to the head noun, these constructions are ambiguous. Whether a certain construction is a subject genitive or an object genitive can only be determined through selectional restrictions and the context of a sentence. According to Lindauer, these functional genitives belong to the class of possessives. They certainly pass the tests mentioned above. But there are two arguments for a more fine grained distinction in these cases. First, it is desirable to avoid, if possible, abstract predicates in the semantic annotation. We would therefore prefer the predicate argument structure in 7 over the one in 8 for the noun phrase *Kolumbus Entdeckung Amerikas* 'Columbus' discovery of America' :

(7) COLUMBUS(x) ∧ AMERICA(y) ∧ DISCOVERY(x,y)

(8) COLUMBUS(x) ∧ AMERICA(y) ∧ DISCOVER(z) ∧ POSS(x,z) ∧ POSS(z,y)

Second, Lindauers analysis doesn't go together with a lexical-functional syntactic approach. In the German "f-structure bank"[4], genitive attributes are treated as grammatical functions. In Lexical Functional Grammar, a uniqueness constraint requires that a grammatical function can appear only once for a certain predicate. However, there is, at least for deverbal nouns, nothing special about two genitive attributes occurring in a German noun phrase. One way to solve the problem is to differentiate between left and right attributes, as Forst [4] did. However, we prefer to subdivide possessive genitives in these cases into subject genitives and adnominal genitives according to Chisarik and Payne's LFG work [2].

Relational nouns The semantic annotation should take into account that the grammatical analysis for relational nouns differs from that of other nouns. The favored predicate argument structure for a noun phrase like *Petras Schwester* 'Petra's sister' is different from the one for a noun phrase like *Petras Pferd* 'Petra's

horse'. The semantic relation is given by the head noun and is therefore directly available. Prototypical for these cases is the kinship relation. The genitive attribute can then be considered a true internal argument in the sense of Barker [1]:

(9) SISTER(x,PETRA)
HORSE(x) ∧ POSS(PETRA,x)

Partitive and possessive genitives Partitive genitives are typically defined semantically: They express part-whole-relations. This is the definition used by Duden and Helbig/Buscha. Lindauer, on the other hand, singles out partitives by the tests mentioned above. In his approach, the notion of partitives is very restricted, and does not cover classic part-whole relations as in *die Augen meiner Freundin* 'the eyes of my friend'. Constructions which mention institutions, as in *der Präsident des Bundesrats* 'the president of the federal council', pose problems for Lindauer's test. A closer look shows that in these cases the genitives express both possessive *and* partitive relations: Paraphrasing (in the style of Helbig/Buscha) gives us both *der Bundesrat hat einen Präsidenten* 'the federal council has a president' and *der Präsident ist ein Teil vom Bundesrat* 'the president is part of the federal council'. For compatibility with more general classifications of semantic relations I decided to tag these genitives separately.

Test Result	**Noun Feature**	**Relation**	**Smultron Label**
Positive	Deverbal	Subject/Adnominal	SUBJ/ADN
	Relational	Kinship	KNS
	Deadjectival	Property	PROP
	Body part	Partitive and Possessive	PP
	–	Possessive	POSS
Inconclusive	Institutions	Partitive and Possessive	PP
	Collocations	Adverbial/–	AVG/–
Negative	Quantity	Partitive	PRT
	Abstract concept	Explicative	EXP

Table 1: Proposed classification scheme and mapping to Smultron labels.

Classification My classification, summarized in Table 1, concentrates on Lindauer's first syntactic test, which states that a possessive genitive may be replaced by a possessive pronoun. The class of possessive genitives (in Lindauer's syntactic sense) is subdivided into six categories depending on whether the head noun is deverbal, deadjectival, or relational. A special category is assigned to nouns denoting body parts. In all other cases, the syntactically possessive genitives are also possessives in the semantic sense. The class of syntactically partitive genitives

is subdivided into real partitives – where the head noun denotes a quantity – and explicative genitives in all other cases. There are some cases for which the test cannot give a definitive answer, namely if an expression involves institutions and in the case of collocations.

4 Annotation in Smultron

I have annotated 392 occurences of genitive attributes in the Smultron corpus. Smultron [8] is a parallel treebank consisting of around 1000 German sentences along with parallel Swedish and English texts, extracted from Jostein Gaarder's novel *Sophie's World* (the literary part) and from company annual reports (the economy part). The German syntax trees are annotated according to the TIGER guidelines. My annotation replaces the edge label 'AG' ("genitive attribute") with one of the ten semantic labels listed above. Interestingly, the use of genitive attributes heavily depends on the text type: There are only 0.10 occurrences per sentence in the literary part, compared to 0.65 occurrences per sentence for the economy part. Sentences containing three and more genitive attributes are common in the latter. First parallel annotations by three annotators showed an inter-annotator

Part	SUBJ	ADN	PROP	KNS	POSS	PP	PRT	EXP	AVG	–
Literary	2	3	0	4	12	0	18	4	8	2
Economy	42	82	0	0	77	60	36	13	19	10

Table 2: Distribution of genitive relations in Smultron

agreement of 88 percent, which we consider a very promising result.

5 Conclusions and Future Work

A closer inspection of the various genitive constructions revealed that their semantic classification directly depends on easily determinable morphologic features of the involved nouns. A combination of Lindauer's replacement test and morphologic analysis of the involved nouns thus enables a detailed semantic classification that has the advantage of being both based on transparent criteria and semantically adequate. A first interesting insight from the annotation of the Smultron Treebank is the surprisingly clear correlation between text type and frequency of genitive attributes.

To get a better idea of the applicability and the quality of the classification scheme described in this paper, it is planned to have other annotators apply it to

the same corpus. It is also planned to analyze the TIGER Treebank. The larger amount of data will enable subsequent experiments on automatic annotation using morphologic analysis of nouns and machine learning methods for classification. Corresponding annotation of the English and Swedish subcorpora of the Smultron Treebank are planned as well.

Acknowledgements I would like to thank Michael Piotrowski, Alexandra Bänzli and Étienne Ailloud. The research presented here was funded by the Swiss National Science Foundation (SNSF).

References

[1] Chris Barker. *Possessive Descriptions*. CSLI Publications, Stanford, 1995.

[2] Erika Chisarik and John Payne. Modelling Possessor Constructions in LFG: English and Hungarian. In *Nominals: Inside and out*. CSLI Publications, 2003.

[3] Dudenredaktion. *Duden: Die Grammatik.* Dudenverlag (= Duden, Band 4), 2005.

[4] Martin Forst. Treebank Conversion. Creating a German f-structure bank from the TIGER Corpus. In *Proceedings of the LFG03 Conference*, 2003.

[5] Roxana Girju, P. Nakov, V. Nastase, S. Szpakowicz, P. Turney, and D. Yuret. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Semantic Evaluation Workshop (SemEval) in conjunction with ACL*, 2007.

[6] Gerhard Helbig and Joachim Buscha. *Deutsche Grammatik. Ein Handbuch fÃ¼r den Ausländerunterricht.* VEB Verlag Enzypklopädie Leipzig, 1984.

[7] Thomas Lindauer. *Genitivattribute*. PhD thesis, Universität Zürich, 1995.

[8] Martin Volk and Yvonne Samuelsson. Frame-Semantic Annotation on a Parallel Treebank. In *Proc. of Nodalida Workshop on Building Frame Semantics Resources for Scandinavian and Baltic Languages*, 2007.

Extracting and Annotating Wikipedia Sub-Domains

— Towards a New eScience Community Resource —

Gisle Ytrestøl[♣], Dan Flickinger[♠], and Stephan Oepen[♣♠]

[♣] University of Oslo, Department of Informatics

[♠] Stanford University, Center for the Study of Language and Information

gisley@ifi.uio.no, danf@stanford.edu, oe@ifi.uio.no

Abstract

We suggest a simple procedure for the extraction of Wikipedia sub-domains, propose a plain-text (human and machine readable) corpus exchange format, reflect on the interactions of Wikipedia markup and linguistic analysis, and report initial experimental results in parsing and treebanking a domain-specific sub-set of Wikipedia content.

1 Motivation and a Long-Term Vision

Linguistically annotated corpora—for English specifically the Penn Treebank (PTB; Marcus, Santorini, & Marcinkiewicz, 1993) and derivatives—have greatly advanced research on syntactic and semantic analysis. However, it has been observed repeatedly that (statistical) parsers trained on the PTB can drop sharply in terms of parsing accuracy when applied to other data sets (Gildea, 2001, inter alios). Ever since its release, there have been concerns about the somewhat idiosyncratic nature of the PTB corpus (primarily Wall Street Journal articles from the late 1980s), in terms of its subject matter, genre, and (by now) age. Furthermore—seeing the cost of initial construction for the PTB, and its still dominant role in data-driven natural language processing (NLP) for English—design decisions made two decades ago perpetuate (sometimes in undesirable ways) into contemporary annotation work. PropBank (Palmer, Gildea, & Kingsbury, 2005), for example, performs semantic annotation on the basis of PTB syntactic structures, such that a discontinuous structure like *Gulf received a takeover bid from Simmons of $50 million.* (simplified from WSJ0178) leads to an analysis of *receive* as a four-place relation (with a dubious ARG4-of role). Quite generally speaking, richly annotated treebanks that exemplify a variety of domains and genres (and of course languages other than

English) are not yet available. And neither are broadly accepted gold-standard representations that adequately support a range of distinct NLP tasks and techniques.

In response to a growing interest in so-called eScience applications of NLP—computationally intensive, large-scale text processing to advance research and education—a lot of current research targets scholarly literature, often in molecular biology or chemistry (Tateisi, Yakushiji, Ohta, & Tsujii, 2005; Rupp, Copestake, Teufel, & Waldron, 2007; inter alios). Due to the specialized nature of these domains, however, many NLP research teams—without in-depth knowledge of the subject area—report difficulties in actually 'making sense' of their data. To make eScience more practical (and affordable for smaller teams), we propose a simple technique of compiling and annotating domain-specific corpora of scholarly literature, initially drawing predominantly on encyclopedic texts from the community resource Wikipedia.[1] Adapting the Redwoods grammar-based annotation approach (Oepen, Flickinger, Toutanova, & Manning, 2004) to this task, we expect to construct and distribute a new treebank of texts in our own field, Computational Linguistics, annotated with both syntactic and (propositional) semantic information—dubbed the WeScience Treebank. Should this approach prove feasible and sufficiently cost-effective, we expect that it can be adapted to additional document collections and genres, ideally giving rise—over time—to an increased repository of 'community treebanks', as well as greater flexibility in terms of gold-standard representations.

In an initial experiment, we gauge the feasibility of our approach and briefly discuss the interaction of (display) markup and linguistic analysis (§ 2 and § 3); we further report on a very preliminary experiment in sentence segmentation, parsing, and annotation (§ 4), and conclude by projecting these into an expected release date for (a first version) of WeScience.

2 Wikipedia—Some Facts and Figures

Wikipedia represents probably the largest and most easily accessible body of text on the Internet. Under the terms of the open-source GNU Free Documentation License, Wikipedia content can be accessed, used, and redistributed freely. Wikipedia to date contains more than 1.74 billion words in 9.25 million articles, in approximately 250 languages. English represents by far the largest language resource, with more than 1 billion words distributed among 2,543,723 articles.[2] Its size, hyper-

[1] In 2007 and 2008, the interest in Wikipedia content for NLP research has seen a lively increase; see `http://www.mkbergman.com/?p=417` for an overview of recent Wikipedia-based R&D, most from a Semantic Web point of view.

[2] Given the highly dynamic nature of Wikipedia, these statistics evolve constantly. We report on the stable 'release' snapshot dated July 2008, which also provides the starting point for our sub-

text nature, and availability make Wikipedia an attractive target for NLP research.

Wikipedia's editing process distinguishes it from most other documents that are available on-line. An article may have countless authors and editors. In April 2008, the English Wikipedia received 220,949 edits a day, with a total of 175,884 distinct editors that month. Wikipedia text provides relatively coherent, relatively high-quality language, but it inevitably also presents a comparatively high degree of linguistic (including stylistic) variation. It is thus indicative of dynamic, community-created content (see below).

2.1 Domain-Specific Selection of Text

Our goal in the WeScience Treebank is to extract a sub-domain corpus, targeting our own research field—NLP. To approximate the notion of a specific sub-domain in Wikipedia (or potentially other hyper-linked electronic text), we start from the Wikipedia category system—an optional facility to associate articles with one or more labels drawn from a hierarchy of (user-supplied) categories. The category system, however, is immature and appears far less carefully maintained than the articles proper. Hence, by itself, it would yield a relatively poor demarcation of a specific subject area.

For our purposes, we chose the category *Computational Linguistics* and all its sub-categories—which include, among others, *Natural Language Processing*, *Data Mining* and *Machine Translation*—to activate an initial seed of potentially relevant articles. Altogether, 355 articles are categorized under *Computational Linguistics* or any of its sub-categories. However, some of these articles seemed somewhat out-of-domain (see below for examples), and several are so-called stub articles or very specific and short, e.g. articles about individual software tools or companies. It was also apparent that many relevant articles are not (yet) associated with either of these categories. To compensate for the limitations in the Wikipedia category system, we applied a simple link analysis and counted the number of cross-references to other Wikipedia articles from our initial seed set. By filtering out articles with a comparatively low number of cross-references, we aim to quantify the significance (of all candidate articles) to our domain, expecting to improve both the recall and precision of sub-domain extraction.

Among the articles that were filtered out from our original set of seed articles, we find examples like *AOLbyPhone* (0 references) and *Computational Humor* (1 reference). New articles, differently categorized, were activated based on this approach. These include quite prominent examples like *Machine learning* (34 references), *Artificial intelligence* (33 references) and *Linguistics* (24 references). Of

domain extraction. See `http://en.wikipedia.org/wiki/Wikipedia` for up-to-date statistics on Wikipedia.

the 355 seed articles, only 30 articles remain in the final selection. Confirming our expectations, filtering based on link analysis eliminated the majority of very narrowly construed articles, e.g. specific tools and enterprises.

However, our link analysis and cross-reference metric also activates a few dubious articles (in terms of the target sub-domain), for example *United States* (8 references). We have deliberately set up our sub-domain extraction approach as a fully automated procedure so far, avoiding any elements of subjective judgment. However, we expect to further refine the results based on feedback from the scientific community.

To suppress the linguistically less rewarding stub articles, we further applied a minimum length threshold (of 2,000 characters, including markup) and—using a minimum of seven incoming cross-references—were left with 100 Wikipedia articles and approximately 270,000 tokens.

2.2 Spelling Conventions and Stylistic Variation

English Wikipedia does not conform to one specific (national) spelling convention, but according to the guidelines for contributors and editors, the language within an article should be consistent with respect to spelling and grammar (thus, *centre* and *center* should not be used side-by-side). Articles are not tagged according to which variant of English they use, and we have yet to gather statistics on the distribution of English variants.

In our view, the WeScience Treebank reflects the kind of content that is growing rapidly on the Internet, namely community-created content. In such content one will typically expect more variation in style (and quality), more spelling mistakes and 'imperfect' grammar, as well as some proportion of text produced by non-native speakers. The WeScience Treebank may provide a useful point of reference for the study of the distribution of such phenomena in community-created content.

3 Wikipedia Markup

Wikipedia articles are edited in the *Wiki Markup Syntax*, a straightforward logical markup language that facilitates on-line rendering (as HMTL) for display in a web browser. Again, there are Wikipedia guidelines and conventions for how to edit or add content, aiming to keep the architecture and design as consistent as possible. In preparing the WeScience Treebank we aim to strike a practical balance between (a) preserving all linguistic content, including potentially relevant markup, and (b) presenting the corpus in a form that is easily accessible to both humans and NLP tools. Thus, we define a textual, line-oriented WeScience exchange format (see

Section 3.2 for a brief discussion of related, XML-based efforts and our rationale for choosing a plain-text format). From the raw source files of Wikipedia articles, we eliminate markup which is linguistically irrelevant—some meta information or in-text image data, for example—but aim to preserve all markup that may eventually be important for linguistic analysis. Markup indicating bulleted lists, (sub)headings, hyper-links, or specific font properties, for example, may signal specialized syntax or use–mention contrasts.

Following are some examples of Wikipedia markup that we want to preserve, because it closely interacts with 'core' linguistic content:

(1) [10120240] |* Design of [[parser]]s or [[phrase chunking|chunkers]]
for [[natural language]]s

(2) [10621290] |For example, in the following example, ″one″
can stand in for ″new car″.

The WeScience Treebank provides gold-standard 'sentence' boundaries (sometimes sentential units are not sentences in the linguistic sense; see below) with unique sentence identifiers. Examples (1) and (2) show the actual WeScience file format, where each sentence is prefixed by its identifier and the '|' separator symbol. In (1), the initial '*' indicates items in a bulleted list (which can exhibit various specialized syntactic patterns) and the square brackets show Wikipedia hyperlinks. Example (2), on the other hand, shows the use of *italics* (the interpretation of the double apostrophe in Wikipedia markup) for the purpose of quoting; i.e. the use–mention distinction is made as a font property only. We further discuss the interactions of display markup and syntactic structures in Section 4.2 below.

3.1 From Source Markup to the WeScience Corpus

Unwanted markup and entire sections from the source articles have been automatically removed, mainly by the use of a large number of regular expressions. These are parts of the articles which, as we see it, are irrelevant to linguistic analysis, including entire sections—like *See Also*, *References*, or *Bibliography*—or links to images, comments made by other users, and various Wikipedia-internal elements.

Once reduced to what we consider (potentially) relevant linguistic content, we applied semi-automated sentence segmentation. In a first, automated step, all linebreaks were removed from the original source text, and the open-source package `tokenizer`[3] was used to insert sentence boundaries. This is a rule-based tool which proved very capable as a sentence segmenter. The procedure was further

[3] See `http://www.cis.uni-muenchen.de/~wastl/misc/` for background and access.

optimized by some customization, based on a manual error analysis. Most of the errors made by the segmenter could be attributed to 'misleading' (remaining) markup in its input (`tokenizer`, by default, expects 'pure' text). For instance, the tool initially failed to insert segment boundaries between some numbered list elements (where the Wikipedia markup '#', in a sense, takes on the function of sentence-initial punctuation). We have augmented our pre-processing pipeline with regular expressions that force segment boundaries in cases where `tokenizer` failed. Furthermore, we are currently experimenting with an additional layer of 'temporary simplification' for the sentence segmentation step. For markup elements that can be asserted to never span segment boundaries (Wikipedia links, `<source>` and `<code>` blocks, for example), segments of original text are temporarily replaced with simplified, placeholder tokens. Once sentence segmentation is complete, these replacements are reverted.

A second round of error analysis on a sub-set of 1,000 segments suggests a residual error rate of about 4 per cent, with half of these text preparation errors due to incomplete handling of wiki mark-up (e.g. for `<math>` and `<code>` blocks, colons marking indentation, and some hyperlinks). The other half of these errors are due to missing or spurious sentence breaks (often due to unusual punctuation clusters), and to confusion of picture captions or section headers with main text. After one more round of tuning of these preparation scripts, we will manually inspect and correct segment boundaries as we treebank.

The WeScience exchange format presents one sentence per line, in (mostly) plain text form. Unique eight-digit sentence identifiers provide ease of reference and are coded in a form that directly points back to the original source article (first three digits). The approximately 270,000 tokens in the corpus in our current selection amount to about 14,000 sentences, ranging up to 185 tokens in length (excluding markup), with a relatively dense distribution up to about 50 tokens (e.g. 107 sentences are between 50 and 55 tokens long). Sentences are consecutively distributed across 16 files ('sections' of a sort), where each section comprises up to 1,000 sentences, and no article is split between two files.

3.2 Related Initiatives

There are of course numerous other NLP initiatives who look at Wikipedia text as a corpus. One such resource is the WikiXML initiative at the Information and Language Processing Systems group of the University of Amsterdam.The project web site summarizes:

> Our XML version of Wikipedia was designed to serve as a multilingual text collection for experiments in Information Retrieval and

> Natural Language Processing, in particular, in the context of Cross-Language Evaluation Forum (CLEF).
>
> (`http://ilps.science.uva.nl/WikiXML/`)

Besides providing XML snapshots of Wikipedia in various languages, WikiXML offers a conversion tool that can be used to convert articles from Wikipedia Markup to XML format. In a sense, this step can be viewed as making article and markup more explicit, forcing it into the more rigid scheme of a validated XML document. The Amsterdam initiative is just one of several projects which convert Wikipedia native format to XML, typically with the explicit or implied goal of easing NLP tasks on Wikipedia text. WikipediaXML (Denoyer & Gallinari, 2006), for example, is another closely related initiative.

Although projects like these clearly pull in the same direction as ours with respect to preparing Wikipedia-based NLP corpora, we have deliberately decided against XML markup and, thus, were left developing our own preprocessing and conversion tools. First, in principle Wikipedia markup is no less explicit than XML, nor are document validation techniques restricted to XML (the Wikipedia server infrastructure, in a sense, validates Wikipedia markup every time it renders an article for on-line browsing). But largely owed to its compactness and design aiming to facilitate source-level editing, we find that Wikipedia markup strikes a better balance between machine and human readability.

Second, and more importantly, the central unit of analysis in our setup is the sentence. For increased flexibility in manipulating WeScience files, we want it to be the case that both individual sentences and any concatenation of sentences are valid structural units. Thus, in the tradition of the Un*x operating system, we have opted for a textual, line-oriented exchange format. In XML, on the other hand, there is no straightforward way of concatenating multiple documents into a new, valid document. In treebanking Wikipedia text, the document-level token structure and linguistic tokenization need not always be compatible. In example (1) above, for example, the sub-string *[[natural language]]s* would likely be considered two document tokens (one being a Wikipedia link, with a juxtaposed string token *s*), but in terms of syntactic structure *languages* will have to be a single word form. Where Wikipedia markup can leave document token structure implicit, our parsing and treebanking machinery (see Section 4) keeps track of linguistic token positions in terms of character position stand-off pointers. Hence, it is possible for *languages* to span a character range that does not coincide with document-level token boundaries.

4 Initial Parsing and Treebanking Results

In annotating the WeScience Treebank, we plan to apply the grammar- and discriminant-based approach of Oepen et al. (2004)—the open-source LinGO Redwoods environment, based on the LinGO English Resource Grammar (ERG; Flickinger, 2000).[4] In the Redwoods approach, the treebank records the complete syntacto-semantic analyses provided by the grammar (in the HPSG framework), coupled with tools to extract different kinds of linguistic representation (at variable granularity)—primarily labeled syntax trees, 'deep' dependency structures, and logical-form meaning representations. Annotation in Redwoods amounts to disambiguation among the candidate analyses proposed by the grammar and, of course, analytical inspection of the final result. To gauge the feasibility (and scalability) of this approach, we performed an initial 'blind' experiment, applying the ERG to the complete WeScience corpus and asking an expert linguist to disambiguate two of the 16 sections. To the best of our knowledge, the ERG has not been previously applied to Wikipedia text, hence it is to be expected that—ideally in joint work with the LinGO team—grammar and parser performance can be further improved.

4.1 An 'Out-of-the-Box' Experiment

For this initial experiment, we constructed another layer of regular expressions (thirteen in total), to simply eliminate Wikipedia markup prior to parsing.[5] Basic parsing coverage of the corpus with the ERG reached 86 percent, with variation of up to 5 per cent in either direction for any one of the 16 sections. For each sentence, we recorded up to 500 highest-ranked analyses, using a pre-existing parse-selection model (for a different domain, viz. hiking instructions). In this initial experiment, we also imposed relatively restrictive resource limits on the time and memory usage permitted for any one sentence (a maximum of one gigabyte in process size or one minute in cpu time). The average number of tokens in each sentence is 17.9, and (using this specific configuration) we observed average parse times per sentence (to produce up to 500 analyses for treebanking) of just below five seconds.

We were positively impressed with comparatively good parsing coverage in this 'out-of-the-box' experiment, applying the ERG to a new genre and subject area.[6]

[4] The Redwoods approach is essentially a scaled-up adaptation of the techniques originally proposed by Carter (1997). Grammar- and discriminant-based annotation has been applied successfully to multiple languages and frameworks, for example by Bouma, van Noord, & Malouf (2001) and Rosén, De Smedt, & Meurer (2006), inter alios.

[5] For reasons discussed briefly in § 3 above, this simple-minded approach may inhibit successful analysis in some cases. In future experiments we aim to collaborate with the ERG developers on a genuine integration of pre-processing and parsing (using the framework of Adolphs et al., 2008).

[6] The grammar includes an unknown word facility, postulating underspecified lexical entries for

From earlier Redwoods reports, however, it is to be expected that basic parse success does not necessarily indicate the existence of a *correct* analysis. To determine the expected proportion of invalid analyses, we manually treebanked (using the Redwoods discriminant machinery) all inputs that parsed from two sections of the 16 sections. This exercise indicated a (relatively high) rejection rate of nearly 30 percent, i.e. during treebanking the annotator only found a correct analysis for a little more than 70 percent of all sentences.[7]

Combining 14 percent parse failures and a 30-percent rejection rate during treebanking, our prelimary experiment arrives at a fully correct analysis for about 60 percent of the segments in the WeScience corpus. These analyses are 'correct' according to the linguistic assumptions of the HPSG framework and its implementation in the ERG, and as such they provide comparatively fine-grained syntacto-semantic information—at moderate annotation cost. To further put these results into perspective, and of course to estimate to what degree one can hope to reduce treebank 'gaps' (introduced by either parse or annotation failure), we applied a first manual error analysis to the same two treebanked sections. This process revealed a number of distinct sources of failure, grouped as either shortcomings in *corpus preparation* or *analysis errors*. Table 1 shows a more detailed break-down of failure types, including a rough estimate (based on our selection of two sections) of the percentage of items affected in the full corpus.

4.2 Integrating Markup and Syntactic Analysis

As pointed out earlier, in some cases markup properties directly affect linguistic analysis. A prominent such example is the use of italics in a function similar to quotation (drawing use–mention distinctions or demarcating foreign language material), as we saw in example (2) above. To abstract from the concrete syntax of a specific markup language (like Wikipedia, HTML, or LaTeX), we have started to augment the ERG analysis grammar with selected elements of an *abstract* Grammatical Markup Language (dubbed GML). During input pre-processing for the parser, for example, italicized words or phrases are enclosed in 'opening' and 'closing italics' tokens: ¦*i new car i*¦, for part of example (2).

Parser-internally, GML tokens like these are treated much like punctuation marks, i.e. in the approach of Adolphs et al. (2008) such tokens are re-combined

open class categories on the basis of a standard PoS tagger. On the other hand, the configuration we used does *not* include additional robustness measures, i.e. the parser will fail in case there is no complete analysis, spanning the full input string.

[7] For comparison, Oepen et al. (2004) report a ten percent rejection rate. However, in their exercise they were treebanking considerably easier text (transcribed task-oriented dialogues), and working in a domain for which the ERG had been carefully adapted already.

	Corpus Preparation
1%	missing sentence breaks, spurious sentence breaks, and lost text
1%	confusion of picture captions or section headers with main text
2%	mark-up handling, such as `<math>` blocks and special environments

	Parsing Errors
4%	resource limitations (cpu time or size of search space)
5%	named-entity and other unknown-word handling problems
10%	parse ranking: good analysis likely available, but not in top 500
15%	grammar shortcomings, e.g. pseudopassives (... *is referred to as*)
1%	text errors: typos, missing determiners, *its* vs. *it's* confusion, etc.
1%	miscellaneous: tagger errors, foreign language illustrations, etc.

Table 1: Distribution of coarse-grained error types, including some examples and estimated overall percentages.

with adjacent 'regular' tokens (i.e. non-markup and non-punctuation ones) and then syntactically analyzed as pseudo-affixes. This approach has the benefit of eliminating attachment ambiguities for punctuation and markup tokens (they always attach lowest, i.e. lexically), and furthermore it yields a perfect predictor of standard whitespace conventions around punctuation marks—commas, for example, are pseudo-suffixes; opening parentheses, on the other hand, are pseudo-prefixes. Aligning the treatment of some markup with the existing analysis of punctuation provides a fruitful starting hypothesis for our WeScience experiments. In the case of italicized phrases, the grammar is thus enabled to apply its existing apparatus for 'recognizing' quoted expressions (as an uninterpreted, strictly left-branching binary tree). Once a complete, properly bracketed phrase is recognized, unary rules map the corresponding constituent into a suitable syntactic category, in this case a proper name.

5 Outlook—The Immediate and Mid-Term Future

We embarked on the WeScience effort primarily to enable our own research (on large-scale semantic parsing of scholarly literature, 'deep' information extraction, and ontology learning). But we believe it is worth documenting our results, even in this early stage, because Wikipedia (and other) sub-domain corpora could develop into valuable NLP resources; particularly so, if we can find ways of extending the Wikipedia 'community' approach from creating the original content to the creation of additional linguistic annotation.

In our view, our very preliminary results are encouraging in multiple ways. The combination of an initial seed of documents (in our case derived from the Wikipedia category system, no matter how immature its current status) and simple link cardinality analysis provides a straightforward way of sub-domain extraction. Seeing remaining fuzziness in the notion of our 'target domain' (i.e. NLP-related articles), we see no objective metric to gauge the success of our initial experiment. But in informal presentations to multiple (external) colleagues, we have found our intuition confirmed that our current WeScience collection exhibts a (much) greater degree of domain coherence than the original seed set.

Pre-processing a richly marked-up hypertext for NLP purposes—finding a good balance between human and machine readability, on the one hand, and preserving anything that potentially has bearing on linguistic content, on the other hand—is a challenging exercise in its own right. Through an early public release of the WeScience corpus, we hope to be able to gather community feedback on this aspect of our proposal.

As regards the application of off-the-shelf NLP tools—sentence segmentation, grammatical analysis, and discriminant-based treebanking—and the interactions of document mark-up and linguistic analysis, our simple-minded experiments seem to suggest both a basic level of feasibility and a substantial remaining potential for improvement. As in most (precision-oriented) NLP, we expect that an iterative feedback loop of in-depth error analysis and careful adaptation of the processing pipeline (both at the corpus creation and parsing layers) will facilitate substantial improvements in segmentation, parsing, and treebanking. This work, in itself, will illuminate relevant linguistic properties of Wikipedia-like content, and of course of the open-source NLP resources involved.

Even with substantial improvements in treebanking coverage, say up to the levels reported by Oepen et al. (2004) (of about 85 percent), to actually make the WeScience Treebank useful as a *treebank*, we will need to address the problem of remaining coverage gaps (i.e. out-of-scope inputs from the ERG perspective; be that owed to actual non-grammaticality or lacking grammatical coverage). We expect to adapt the robust parsing approach of Zhang & Kordoni (2008), extend it to facilitate robust meaning composition, and integrate it in the Redwoods environment with a novel facility for post-editing (this is similar to the techniques used in the construction of the Alpino treebank; Bouma et al., 2001).

The success of open-source projects and particularly the Linux operating system is at times attributed to the *release early, release often* paradigm used successfully to coordinate the efforts of distributed communities of developers. In the hope that our WeScience efforts may stimulate adaptation by others, we plan to

release a first version in early 2009.[8] This version will minimally provide a stable selection of in-domain articles and gold-standard sentence segmentation. A complete, treebanked version—though excluding a residual percentage of out-of-scope items—will be made available in the first half of 2009. In joint work with the LinGO developers, we expect to adapt both the grammar (extend or improve linguistic analyses) and parsing technology in the light of the WeScience experience.

References

Adolphs, P., Oepen, S., Callmeier, U., Crysmann, B., Flickinger, D., & Kiefer, B. (2008). Some fine points of hybrid natural language parsing. In *Proceedings of the 6th International Conference on Language Resources and Evaluation.* Marrakech, Morocco.

Bouma, G., van Noord, G., & Malouf, R. (2001). Alpino. Wide-coverage computational analysis of Dutch. In W. Daelemans, K. Sima-an, J. Veenstra, & J. Zavrel (Eds.), *Computational linguistics in the Netherlands* (pp. 45–59). Amsterdam, The Netherlands: Rodopi.

Carter, D. (1997). The TreeBanker. A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering.* Madrid, Spain.

Denoyer, L., & Gallinari, P. (2006). The Wikipedia XML corpus. In *Proceedings of the Fifth Workshop of the Initiative for the Evaluation of XML Retrieval.* Dagstuhl, Germany.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6 (1)*, 15–28.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 conference on Empirical Methods in Natural Language Processing.* Pittsburgh, PA.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, *19*, 313–330.

Oepen, S., Flickinger, D., Toutanova, K., & Manning, C. D. (2004). LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, *2*(4), 575–596.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank. A corpus annotated with semantic roles. *Computational Linguistics*, *31*(1), 71–106.

[8] This initial release of the corpus and partial treebank will be made available on the WeScience home page: `http://www.delph-in.net/wescience/`.

Rosén, V., De Smedt, K., & Meurer, P. (2006, December). Towards a toolkit linking treebanking and grammar development. In *Proceedings of the 5th Workshop on Treebanks and Linguistic Theories* (pp. 55–66). Prague, Czech Republic.

Rupp, C., Copestake, A., Teufel, S., & Waldron, B. (2007). Flexible interfaces in the application of language technology to an eScience corpus. In *Proceedings of the UK eScience Programme All Hands Meeting.* Nottingham, UK.

Tateisi, Y., Yakushiji, A., Ohta, T., & Tsujii, J. (2005). Syntax annotation for the GENIA corpus. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing* (pp. 222–227). Jeju, Korea.

Zhang, Y., & Kordoni, V. (2008). Robust parsing with a large HPSG grammar. In *Proceedings of the 6th International Conference on Language Resources and Evaluation.* Marrakech, Morocco.