Will a Parser Overtake Achilles? First experiments on parsing the Ancient Greek Dependency Treebank

Francesco Mambrini∗ and Marco Passarotti†
∗University of Cologne (Germany)
†Università Cattolica del Sacro Cuore, Milan (Italy)

Abstract

We present a number of experiments on parsing the Ancient Greek Dependency Treebank (AGDT), i.e. the largest syntactically annotated corpus of Ancient Greek currently available (350k words ca). Although the AGDT is rather unbalanced and far from being representative of all genres and periods of Ancient Greek, no attempt has been made so far to perform automatic dependency parsing of Ancient Greek texts. By testing and evaluating one probabilistic dependency parser (MaltParser), we focus on how to improve the parsing accuracy and how to customize a feature model that fits the distinctive properties of Ancient Greek syntax. Also, we prove the impact of genre and author diversity on parsing performances.

1 Introduction

Among the languages currently spoken in the world, Greek has one of the longest documented histories. The first texts written in Greek that have survived to our days date from the mid of the second millennium BCE (around 1420-1400). From the phase that is commonly known as Ancient Greek (9th Century BCE - 6th Century CE), a vast literature has been preserved and thoroughly studied. In rough numbers, two of the most important digital collection of Ancient Greek texts, which are far from being exhaustive, the Thesaurus Linguae Graecae (TLG)[1] and the Perseus Digital Library[2], contain respectively more than 105 and 13 million words.

1.1 Annotated corpora of Ancient Greek

Some of the Ancient Greek texts are today included in two different treebanks.

The Ancient Greek Dependency Treebank (AGDT) [3] is a dependency-based treebank of literary works of the Archaic and Classical age published by Perseus[3].

[1] TLG: `http://www.tlg.uci.edu/`, which goes so far as to the fall of Byzantium (1453 BCE).
[2] Perseus Digital Library: `http://www.perseus.tufts.edu/hopper/`
[3] AGDT: `http://nlp.perseus.tufts.edu/syntax/treebank/greek.html`

In its theoretical framework and guidelines, the AGDT is inspired by the analytical layer of annotation of the Prague Dependency Treebank of Czech [5]. Currently, the last published version of AGDT (1.6) includes 346,813 tokens.

The collection is constituted by unabridged works only: four major poets (Homer, Hesiod, Aeschylus, Sophocles)[4], belonging to two literary genres (epic poetry – Homer, Hesiod – and tragedy – Aeschylus and Sophocles), are represented, as well as one single work of philosophical prose (the *Euthyphro* by Plato). Chronologically, the texts range from the 8th to the late 5th Century BCE. The composition of the AGDT 1.6 is resumed in table 1.

Author	Work	Tokens
Aeschylus	Agamemnon	9,806
	Eumenides	6,380
	Libation Bearers	6,563
	Prometheus Bound	7,064
	Seven Against Thebes	6,206
	Suppliants	5,949
Hesiod	Shield of Heracles	3,834
	Theogony	8,106
	Works and Days	6,941
Homer	Iliad	128,102
	Odyssey	104,467
Sophocles	Ajax	9,474
	Women of Trachis	8,811
	Electra	10,458
	Antigone	8,716
	Oedipus King	9,746
Plato	Euthyphro	6,097
Total		346,813

Table 1: AGDT 1.6: Composition

The Pragmatic Resources in Old Indo-European Languages corpus (PROIEL) [6], on the other hand, is a multilingual parallel corpus of translations of the New Testament in a selection of Indo-European languages; the Greek section includes also other prose texts of different periods (four books of Herodotus' *Histories*, 5th Century BCE, and Palladius' *Historia Lausiaca*, 5th Century CE). The syntactic

[4]For Hesiod and Aeschylus, the AGDT includes the *opera omnia* of the integrally preserved works (fragments are excluded). Of Sophocles' 7 extant tragedies, 5 are annotated. A tradition that dates from the Antiquity and was followed by convention in the AGDT indicates the legendary figure of Homer as the author of *Iliad* and *Odyssey* (along with other minor compositions); the real existence of one single author for both poems has been denied by the modern Homeric scholarship.

annotation is also based on a dependency grammar, and it is partially inspired by the AGDT. The total of the Ancient Greek annotated data is presently 153,730 tokens[5].

1.2 Open questions and methodology

So far, the annotation of both treebanks has been performed manually. The two collections started by annotating some of the most important texts for current university curricula in Classics. Both on account of the difficulty of the works, which require a hard linguistic and philological training, and of the pedagogical value that can be attached to manual annotation, no use of NLP techniques has been made so far. With the partial exception of [7][6], no comprehensive study has been dedicated to evaluate and improve the performance of parsers on Ancient Greek.

Yet, the available syntactically annotated corpora cover only a small portion of the attested documents, in terms of quantity as well as of representativeness of the different genres and chronological phases. The vast majority of the prose production is not included in the AGDT. Moreover, even for the two genres that are adequately represented, a substantial number of texts still remain to be annotated; apart from the missing 2 tragedies of Sophocles, this is the case with the 19 plays of Euripides (170,118 words, 5th Century BCE) or, for epic poetry, the *Argonautica* of Apollonius Rhodius (45,478 words, 3rd Century BCE) or the so-called *Homeric Hymns* (18,211 words, traditionally attributed to Homer, ranging chronologically from the 7th Century BCE to Late Antiquity).

An efficient dependency parser for Ancient Greek is thus a major acquisition supporting the creation of a balanced annotated corpus.

A previous study on Latin treebanks [12], which share a number of features with our collections, has shown that genre and chronology have a decisive influence on the accuracy of different parsers. Three questions then appear to be relevant for a preliminary study:

1. what are the parser's settings ("feature model") that fit best the AGDT?
2. what is the impact of genre and author on the parser's performances?
3. following 2, how should a training set be built? Is the size more relevant than data homogeneity in terms of genre and author?

In this paper, we provide a first answer to these three questions by studying the performances of one single dependency parser, which is trained and tested on different sections of our corpus. From the top-ranking list of the CoNLL-X shared task[7], we selected MaltParser [10], on account of its flexibility, in terms of

[5] This total was communicated to us by the general editor of the project. As the annotation process is still in progress, the numbers change every day.

[6] By leveraging a Greek-Hebrew parallel corpus, [7] develops a target-specific method to improve the parsing accuracy of the Greek Old Testament.

[7] `http://ilk.uvt.nl/conll/`

both parsing algorithm and feature setting. For the task of algorithm and feature selection, we used MaltOptimizer [2] as a starting point, whose output we have evaluated and further improved[8].

1.3 The data

Our data were taken from the AGDT and converted to the CoNLL format[9].

The texts taken from the AGDT were organized by author and genre, in order to evaluate how MaltParser performs with varying authors and genres. It is well known [11] that non-projectivity crucially affects the efficiency of dependency parsers. In comparison with the treebanks used in CoNLL-X [4, 155, tab. 1] and CoNLL 2007 shared tasks [9, 920, tab. 1], our data show a remarkably higher rate of non-projective arcs.

The subsets that we used, along with the number and percentage of non-projective arcs, are resumed in table 2[10].

Data set	Works/Authors	Sentences	Tokens	Non-proj. arcs	%
Homer	Il., Od.	15175	232569	62013	26.66
Tragedy	Aesch., Soph.	7897	95363	21747	22.80
Sophocles	Soph.	3873	47205	10456	22.15

Table 2: Data sets

Each of the subsets in table 2 was randomly partitioned into 5 training and testing sets, all in a ratio of approximately 9:1, in order to perform 5 different experiments.

We first focused on the test sets of Homer (table 3) and Sophocles (table 4).

Then, we studied how genre and author affect MaltParser in detail. We used a model trained on the Homeric poems to evaluate 3 different subsets: (a) the whole annotated work of Hesiod (18,881 tokens: same genre as the training set, but different author), (b) a sample from Sophocles of roughly the same size as Hesiod (18,418 tokens: different genre and author), and (c) the whole available Plato (6,091 tokens: different genre, different author, prose text).

[8] In all our experiments, we used LIBSVN as learning algorithm for MaltParser.

[9] The CoNLL format includes the following 10 fields, although only the first 8 contain non-dummy values: ID (token counter), FORM, LEMMA, CPOSTAG (coarse-grained PoS), POSTAG (fine-grained PoS), FEATS (unordered set of morphological features), HEAD (head of current token, i.e. a value of ID), DEPREL (dependency relation to the HEAD); `http://nextens.uvt.nl/depparse-wiki/DataFormat`

[10] Since not all of the tragedies of Sophocles were already published at the time when we started our work, we used an unfinished version of the *Oedipus King*: 1361 tokens from the ca. last 100 lines of the play were unannotated.

Set Name	Sentences	Tokens	% Train/Test
Homer_Test1	1686	25898	11.14
Homer_Test2	1562	23258	10.00
Homer_Test3	1469	23267	10.00
Homer_Test4	1500	23274	10.01
Homer_Test5	1486	23259	10.00

Table 3: Homer: Test sets

Set Name	Sentences	Tokens	%Train/Test
Sophocles_Test1	430	5186	10.99
Sophocles_Test2	389	4725	10.01
Sophocles_Test3	389	4726	10.01
Sophocles_Test4	384	4731	10.02
Sophocles_Test5	386	4721	10.00

Table 4: Sophocles: Test sets

2 Results and evaluation

2.1 Algorithm and feature selection

Not surprisingly, given the above reported non-projective rates in our data sets, MaltParser scores rather poorly with the default algorithm (Nivre, a linear-time algorithm limited to projective dependency structures) and model (Arceager). With this configuration, training and testing the parser on the Homeric poems (tab. 3), we attained the following results (baseline): 44.1% LAS, 60.3% UAS, 49.2% LA [4][11].

By applying the options suggested by MaltOptimizer, we increased the accuracy of the parser considerably. Due to the high number of non-projective trees and of nodes attached to the root, MaltOptimizer recommends the adoption of the following options[12]:

- adoption of a non-projective algorithm: Covington non-projective is suggested;
- use of the label "AuxK" (terminal punctuation) as default for unattached

[11] The metrics used are the following: Labeled Attachment Score (LAS): the percentage of tokens with correct head and relation label; Unlabeled Attachment Score (UAS): the percentage of tokens with the correct head; Label Accuracy (LA): the percentage of tokens with the correct relation label.

[12] For a quick introduction to MaltParser optimization, see [8]; for a detailed explanation of each option see [1].

tokens that are attached to the technical root node;

- covered root set to "left" (see [8, 7]);
- "shift" allowed: the parser is allowed to skip remaining tokens before the next target token is shifted to the top of the stack;
- root node treated as a token: root dependents are allowed to be attached with a RightArc transition.

The feature model suggested by MaltOptimizer is reported in Appendix, tab. 10.

Starting from these customized options for the Covington non-projective algorithm, we modified the feature model, according to our knowledge of both Ancient Greek and the annotation style of AGDT. We tested and evaluated 6 different configurations: the best performances were attained with experiment n. 4 (Exp4).

The feature model was then modified according to the requirements of the other non-projective algorithm (Stacklazy), so as to evaluate the parser with both non-projective algorithms available for MaltParser[13]. Then, we compared the performances using two different training and test sets: Homer (trained on Homer) and Sophocles (trained on the Tragedy training set: see sec. 2.2 for the choice). Table 5 lists the results of these experiments with MaltOptimizer configuration (MO) and with our feature models for both algorithms (Exp4), all compared to the baseline (first line of tab. 5).

Test set	Training	Algorithm	Feature mod.	LAS	UAS	LA
Homer	Homer	nivreager	arceager	44.1	60.3	49.2
Homer	Homer	covnonproj	MO	69.02	76.46	78.66
Homer	Homer	covnonproj	Exp4	70.96	77.9	80.34
Homer	Homer	stacklazy	Exp4	71.72	78.26	81.62
Soph.	Tragedy	covnonproj	MO	55.24	64.12	67.48
Soph.	Tragedy	covnonproj	Exp4	57.7	65.52	70.14
Soph.	Tragedy	stacklazy	Exp4	56	63.92	69.12

Table 5: Evaluation of algorithms and models: Average of 5 experiments

The configuration Exp4 improves the accuracy for all the metrics. Covington's algorithm surpasses Stacklazy with the corpus of the tragic poems, but the opposite is true in the case of *Iliad* and *Odyssey*.

We first evaluated the accuracy of the parser by grouping the results of the best scoring configuration (Exp4) by dependency relations (tab. 6). The main depen-

[13]The modified feature models are reported in the Appendix, tab. 11 (Covington) and tab. 12 (Stacklazy).

dency relations (PRED, OBJ, SBJ, ADV, ATR, PNOM)[14] attain a good level of accuracy in both subsets, while the performances decrease considerably whenever coordination is concerned.

	Homer		Sophocles	
DepRel	Cov	St	Cov	St
ADV	77.48	75.12	58.94	55.78
ADV_CO	39.32	38.92	18.56	7.36
ATR	75.52	74.84	62.38	64.96
ATR_CO	39.04	39.88	16.28	12.86
AuxC	63.48	57.78	47.96	41.56
AuxP	77.7	73.72	58.72	57.82
OBJ	79.44	76.08	61.26	59.36
OBJ_CO	57.98	63.18	27.36	28.56
PNOM	69.44	68.46	38.28	32.04
PNOM_CO	36.48	36.68	1.82	1.82
PRED	83.52	81.56	85.6	74.8
PRED_CO	61.7	78.78	32.42	43.78
SBJ	82.96	81.5	64.74	58.86
SBJ_CO	58.48	61.26	11.64	6.1

Table 6: Accuracy (LAS) grouped by DEPREL

Non-projectivity's impact on results is quite strong. As reported in table 7, accuracy is considerably lower in case of non-projective relations.

Finally, we grouped the results by PoS tag. Fig. 1 show the different performances for each of the 13 part-of-speech labels used in the AGDT.

[14]The tag PRED is given to the predicate of the main clause of a sentence. An ATR is a sentence member that further specifies a noun in some respect; typical attributives are adjectives, relative clauses and nouns in the genitive case. The difference between OBJ and ADV roughly corresponds to the one between arguments and adjuncts of verbs or adjectives. SBJ: subject. PNOM: nominal predicate. AuxP: prepositions; AuxC: conjunctions. In the event that a node is member of a coordinated construction, the DEPREL label is appended with the suffix _Co.

	Homer		Sophocles	
Projectivity	Cov	St	Cov	St
Projective	72.18	73.32	60.28	58.34
Non-proj.	60.84	58.46	36.24	36.5

Table 7: Accuracy (LAS) grouped by arc projectivity

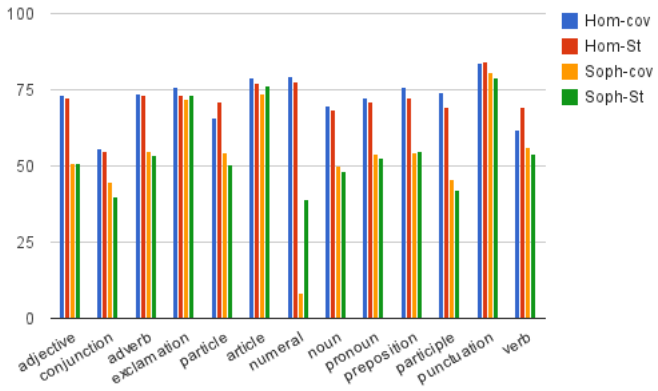

Figure 1: Accuracy (LAS) grouped by PoS tag

2.2 Differences in genre and author

In order to evaluate the role of genre and author diversity and the composition of the best training set both in terms of accuracy of the results and of computational costs, we performed two different experiments.

Firstly, we parsed the 5 sets of approximately 5,000 tokens of Sophocles (tab. 4) using our best performing configuration (Exp4 with Covington non-projective) and with models trained on: (a) the Sophocles training set, (b) all the available tragedies, and (c) the whole AGDT. The average results are reported in table 8.

Training	LAS	UAS	LA	Learning Time
Sophocles	56.14	64.32	68.82	09:20
Tragedy	57.7	65.52	70.14	42:15
AGDT	57.65	66	70	14:26:42

Table 8: Accuracy (LAS) for Soph. with different training sets

As it can be seen, adding more data from texts belonging to the same genre (i.e. tragedy) improves the parser's performances sensibly. Training a model on the whole available treebank slightly decreases the accuracy of LAS and LA, and furthermore, at the cost of a disproportionate growth of learning time. It seems that, in order to parse a text of an author like Sophocles, building a training set from works of the same genre is the most rewarding strategy.

The importance of genre homogeneity is confirmed also by the second experiment we performed. Using our best performing feature models (Exp4 for Coving-

ton and Stacklazy), we trained a model on the available epic poems of Homer and we used it to parse the whole work of the epic poet Hesiod, a sample of roughly equivalent size taken from the tragedies of Sophocles and the only philosophic dialogue of Plato that is included in the AGDT. The results are reported in table 9.

	Cov			Stack		
Test set	LAS	UAS	LA	LAS	UAS	LA
Hesiod	60.7	69.3	72	60.9	68.8	73.7
Sophocles	48.48	58.72	61.3	46.84	56	61.24
Plato	47.1	60.7	60.1	48.2	60.2	62.9

Table 9: Different authors with models trained on Homer

Hesiod's text, the one which is closer to the language of the poems used in the training set both chronologically and for genre, performs far better than the two others. Plato's set is considerably smaller, but it seems that a prose text like a philosophical dialogue is more difficult to parse for a parser trained on epic poetry than a complex poetic text like the one of Sophocles. This result confirms the well known fact that the Homeric poems are a fundamental model and a source of inspiration for the language of Greek poetry. Further studies will be required as soon as new prose texts are added to the collection, in order to confirm this result.

3 Conclusion and future work

We have focused on the performances of a probabilistic dependency parser (MaltParser) on Ancient Greek literary texts. By tuning a series of features, we have considerably improved the efficiency of the parser.

In the process, we have used MaltOptimizer and further improved the feature model suggested by the software, by using the lemmata of the words (in addition to forms) and introducing other modifications that are resumed in tab. 11 and 12.

From our experiments, it emerged that, in order to parse a given Ancient Greek literary work, texts that belong to the same genre as the target text should be used as a training set, rather than the totality of the available collection. This is proved in the case of Sophocles, whose work is not yet fully annotated. Our results can help in providing a more accurate and reliable basis for semi-automatic annotation of the remaining texts of this author and, arguably, for Euripides' work as well.

In the future, we will test other parsers, in order to compare their performances and see whether we can further improve our preliminary results. In particular, we intend to test: DeSR, ISBN, MST, Anna (Mate-Tools)[15]. We will also include

[15]DeSR: `https://sites.google.com/site/desrparser/`; ISBN: `http://cui.unige.ch/~titov/idp/`; MST: `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`; Anna: `http://code.google.com/p/mate-tools/`.

prose texts from the PROIEL corpus, in order to improve the performances on prose texts[16].

Appendix: Feature Models

The feature models used for the experiments are reported in the following tables. The lines of the tables are grouped by Feature Function and Column Name. For an explanation of Feature Functions, Address Functions (both parsing-algorithm-specific functions and dependency-graph functions), as well as of the syntax of the feature-model files, see [1].

Values of FEATS are split by "|" in Left[0], Left[1] for MO; in Left[0], Left[1], Right[0] for Exp4-Cov; in Stack[0], Stack[1], Stack[2], Stack[3] for Exp4-Stack.

Feature Function	Column Name	Address Function
InputColumn	FEATS	Left[0]; Right[0]
InputColumn	FORM	Left[0]; Right[0]; Right[1]; head(Left[0])
InputColumn	POSTAG	LeftContext[0] ;Left[0]; Left[1]; RightContext[0]; Right[0]; Right[1]; Right[2]; Right[3]
OutputColum	DEPREL	Left[0]; Right[0]; ldep(Left[0]); ldep(Left[0]); ldep(Right[0]); rdep(Left[0])

Table 10: MO feature model

References

[1] MaltParser user guide. http://www.maltparser.org/userguide.html.

[2] M. Ballesteros and J. Nivre. MaltOptimizer: a system for MaltParser optimization. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 12)*, Istanbul, 2012. European Language Resources Association (ELRA).

[16]We would like to thank J. Nivre and M. Ballestreros for their kind support with Malt and MaltOptimizer. Professor R. Förtsch and R. Krempel (CoDArchLab, Cologne) also provided decisive help with the infrastructures of the laboratory for digital archaeology at the University of Cologne.

Feature Function	Column Name	Address Function
InputColumn	FEATS	Left[0]; Left[1]; Right[0].
InputColumn	FORM	LeftContext[0]; LeftContext[1]; LeftContext[2]; Left[0]; Left[1]; Left[2]; RightContext[0]; Right[0].
InputColumn	LEMMA	LeftContext[0]; LeftContext[1]; LeftContext[2]; LeftContext[3]; Left[0]; Left[1]; Left[2]; Left[3]; RightContext[0]; RightContext[1]; RightContext[2]; Right[0]; Right[1]; Right[2]; Right[3]; head(Left[0]).
InputColumn	POSTAG	LeftContext[0]; LeftContext[1]; LeftContext[2]; LeftContext[3]; Left[0]; Left[1]; Left[2]; Left[3]; RightContext[0]; RightContext[1]; RightContext[2]; Right[0]; Right[1]; Right[2]; Right[3]; head(Left[0]),
OutputColumn	DEPREL	Left[0]; Right[0]; ldep(Left[0]); ldep(Right[0]); rdep(Left[0]).

Table 11: Exp4-Cov feature model

Feature Function	Column Name	Address Function
InptuColumn	FEATS	Stack[0]; Stack[1].
InptuColumn	FORMS	Input[0]; Lookahead[0]; Stack[0]; Stack[1]
InputColumn	LEMMA	Input[0]; Lookahead[0]; Lookahead[1]; Lookahead[2]; Lookahead[3]; Stack[0]; Stack[1]; Stack[2].
InputColumn	POSTAG	Input[0]; Lookahead[0]; Lookahead[1]; Lookahead[2]; Lookahead[3].
InputColumn	POSTAG	Stack[0]; Stack[1]; Stack[2].
OutputColumn	DEPREL	Stack[0]; Stack[1]; ldep(Stack[0]); ldep(Stack[1]); rdep(Stack[0]); rdep(Stack[1]).

Table 12: Exp4-Stack feature model

[3] D. Bamman, F. Mambrini, and G. Crane. An ownership model of annotation: The Ancient Greek Dependency Treebank. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT 8)*, pages 5–15, Milan, 2009. EDUCatt.

[4] S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X '06)*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[5] A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. The Prague Dependency Treebank: A three-level annotation scenario. In A. Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 103–127. Kluwer, Boston, 2001.

[6] D. Haug and M.L. Jøhndal. Creating a parallel treebank of the old Indo-European Bible translations. In *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*, pages 27–34, Marrakech, 2008. European Language Resources Association (ELRA).

[7] J. Lee. Dependency parsing using prosody markers from a parallel text. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*, pages 127–39, Tartu, 2010. Northern European Association for Language Technology (NEALT).

[8] J. Nivre and J. Hall. Quick guide to MaltParser optimization. http://www.maltparser.org/guides/opt/quick-opt.pdf.

[9] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[10] J. Nivre, J. Hall, and J. Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219, Genoa, 2006. European Language Resources Association (ELRA).

[11] J. Nivre and J. Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[12] M. Passarotti and P. Ruffolo. Parsing the *Index Thomisticus* Treebank. Some preliminary results. In P. Anreiter and M. Kienpointner, editors, *Latin Linguistics Today. Akten des 15. Internationalen Kolloquiums zur Lateinischen Linguistik, Innsbruck, 4. -9. April 2009*, volume 137, pages 714–725, Innsbruck, 2010. Institut für Sprachwissenschaft der Universität Innsbruck.