Autobank: a semi-automatic annotation tool for developing deep Minimalist Grammar treebanks

John Torr
School of Informatics
University of Edinburgh
11 Crichton Street, Edinburgh, UK
john.torr@cantab.net

Abstract

This paper presents Autobank, a prototype tool for constructing a wide-coverage Minimalist Grammar (MG) (Stabler, 1997), and semi-automatically converting the Penn Treebank (PTB) into a deep Minimalist treebank. The front end of the tool is a graphical user interface which facilitates the rapid development of a seed set of MG trees via manual reannotation of PTB preterminals with MG lexical categories. The system then extracts various dependency mappings between the source and target trees, and uses these in concert with a non-statistical MG parser to automatically reannotate the rest of the corpus. Autobank thus enables deep treebank conversions (and subsequent modifications) without the need for complex transduction algorithms accompanied by cascades of ad hoc rules; instead, the locus of human effort falls directly on the task of grammar construction itself.

1 Introduction

Deep parsing techniques, such as CCG parsing, have recently been shown to yield significant benefits for certain NLP applications. However, the construction of new treebanks for training and evaluating parsers using different formalisms is extremely expensive and time-consuming. The Penn Treebank (PTB) (Marcus et al., 1993), for instance, the most commonly used treebank within NLP, took a team of linguists around three years to develop. Its structures were loosely based on Chomsky's Extended Standard Theory (EST) from the 1970s and, nearly half a century on, these look very different from contemporary Chomskyan Minimalist analyses. Considerable theoretical advances have been made during that time, including the discovery of many robust cross-linguistic generalizations. These could prove very useful for NLP applications such as machine translation, particularly with respect to under-resourced languages. Unfortunately, the lack of any Minimalist treebank to date has meant that there has been very little research into statistical Minimalist parsing (though see Hunter and Dyer (2013)).

Given the labour intensity of constructing new treebanks from scratch, computational linguists have developed techniques for converting existing treebanks into different formalisms (e.g. Hockenmaier and Steedman (2002), Chen et al. (2006)). These approaches generally involve two main subtasks: the first is to create a general algorithm to translate the existing trees into the representational format of the target formalism, for example by binarizing and lexicalizing the source trees and, in the case of CCGbank (Hockenmaier and Steedman, 2002), replacing traces of movement with alternative operations such as type-raising and composition; the second task involves coding cascades of ad hoc rules to non-trivially modify and/or enrich the underlying phrase structures, either because the target formalism requires this, or because the researcher disagrees with certain theoretical decisions made by the original treebank's annotators. CCGbank, for instance, replaces many small clauses in the PTB by a two-complement analysis following Steedman (1996).

Autobank is a new approach to semi-automatic treebank conversion. It was designed to avoid the need for coding complex transduction algorithms and cascades of ad hoc rules. Such rules become far less feasible (and difficult for future researchers to modify) when transducing to a very deep formalism such as a Minimalist Grammar (MG) (Stabler, 1997), whose theory of phrase

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 81–86
©2017 Association for Computational Linguistics

structure[1] differs considerably from that of the PTB[2]. Furthermore, given the many competing analyses for any given construction in the Minimalist literature, no single MG treebank will be universally accepted. It is therefore hoped that Autobank will stimulate wider interest in broad coverage statistical Minimalist parsing by providing researchers with a relatively quick and straightforward way to engineer their own MGs and treebanks, or to modify existing ones.

Autobank works as follows: the PTB first undergoes an initial preprocessing phase. Next, the researcher builds a seed MG corpus by annotating lexical items on PTB trees with MG categories and then selecting from among a set of candidate parses which are output using these categories by MGParse, an Extended Directional Minimalist Grammar (EDMG) parser (see Torr and Stabler (2016)). The system then extracts various dependency mappings between the source and target trees. Next, a set of candidate parses is generated for the remaining trees in the PTB, and these are scored using the dependency mappings extracted from the seeds. In this way, the source corpus effectively adopts a disambiguation role in lieu of any statistical model. The basic architecture of the system, which was implemented in Python and its Tkinter module, is given in fig 1.

2 Preprocessing

Autobank includes a module for preprocessing the PTB which corrects certain mistakes and adds some additional annotations carried out by various researchers since the treebank's initial release. For example, following Hockenmaier and Steedman (2002), we have corrected cases where verbs were incorrectly labelled with the past tense tag VBD instead of the past participle tag VBN. We also extend the PTB tag set to include person, number and gender[3] information on nominals and pronominals, in order to constrain reflexive binding and agreement phenomena in MGbank.

The semantic role labels of PropBank (Palmer et al., 2005) and Nombank (Meyers et al., 2004) have also been added onto PTB non-terminals[4], along with the head word and its span. For instance, the AGENT subject NP *Jack* in the sentence, *Jack helped her*, would be annotated with the tag ARG0`{helped<1,2>}`. We have also added the additional NP structure from Vadas and Curran (2007), the additional structure for hyphenated compounds included in the Ontonotes 5 (Weischedel et al., 2012) version of the PTB, and the additional structure and role labels for coordination phrases recently released by Ficler and Goldberg (2016). For all structure added, any function tags are redistributed accordingly[5].

Finally, PropBank includes additional antecedent-trace co-indexing which we have also imported, and some of this implies the need for additional NP structure beyond what Vadas and Curran have provided. For instance, in the phrase, *the unit of New York-based Lowes Corp that *T* makes kent cigarettes*, the original annotation has *the unit* and *of New York-based Lowes Corp* as separate sister NP and PP constituents (with an SBAR node sister to both), both of which are co-indexed with the subject trace (**T**) position in PropBank. In such cases we have added an additional NP node resolving the two constituents into a single antecedent NP.

3 The manual annotation phase

Autobank provides a powerful graphical user interface enabling the researcher to construct an (ED)MG by relabelling PTB preterminals with MG categories and selecting the correct MG parse from a set of candidates generated by the parser.

The main annotation environment is shown in fig 2. The PTB tree and its MG candidates are respectively displayed on the top and bottom of the screen. Between these are a number of buttons allowing for easy navigation through the PTB, including a regular expression search facility for locating specific construction types by searching both the bracketing and the string. The user can also choose to focus on sentences of a given string length. On the left, the sentence is displayed from top to bottom, each word with a drop-down menu listing all MG categories so far associated with that word's PTB preterminal category. Like

[1] MGs are a mildly context sensitive and computational interpretation of Chomsky's (1995) Minimalist Program.

[2] For example, Minimalist trees contain many more null heads and traces of phrasal movement, along with shell and Xbar phrase structures, cartographic clausal and nominal structures, functional heads, head movements, covert movements/Agree operations etc.

[3] We used the NLTK name database to derive gender on proper nouns.

[4] Among other things, these crucially distinguish adjuncts from arguments, raising/ECM from subject/object control, and *promise*-type subject control from object control/ECM.

[5] We use a modified version of Collins' (1999) head finding rules for this task as well as for the dependency extraction.

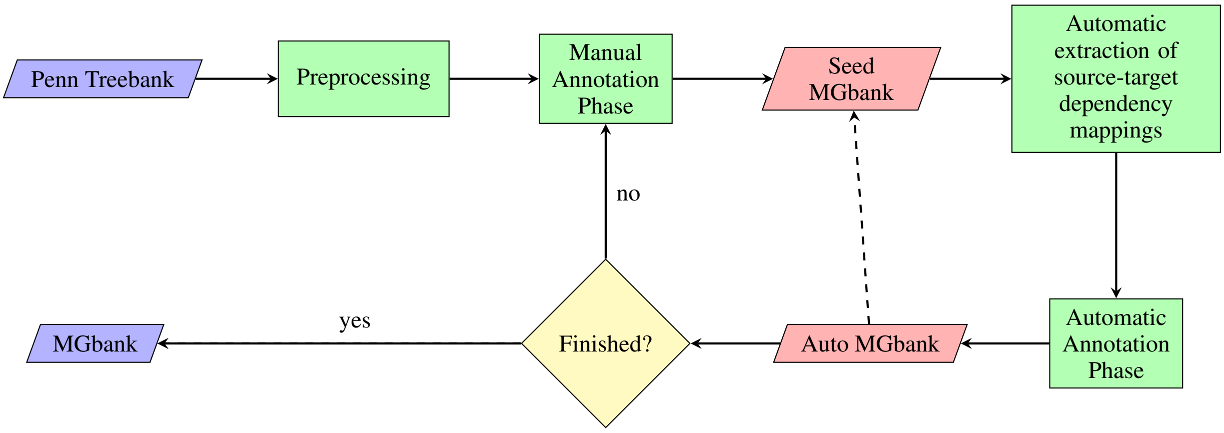
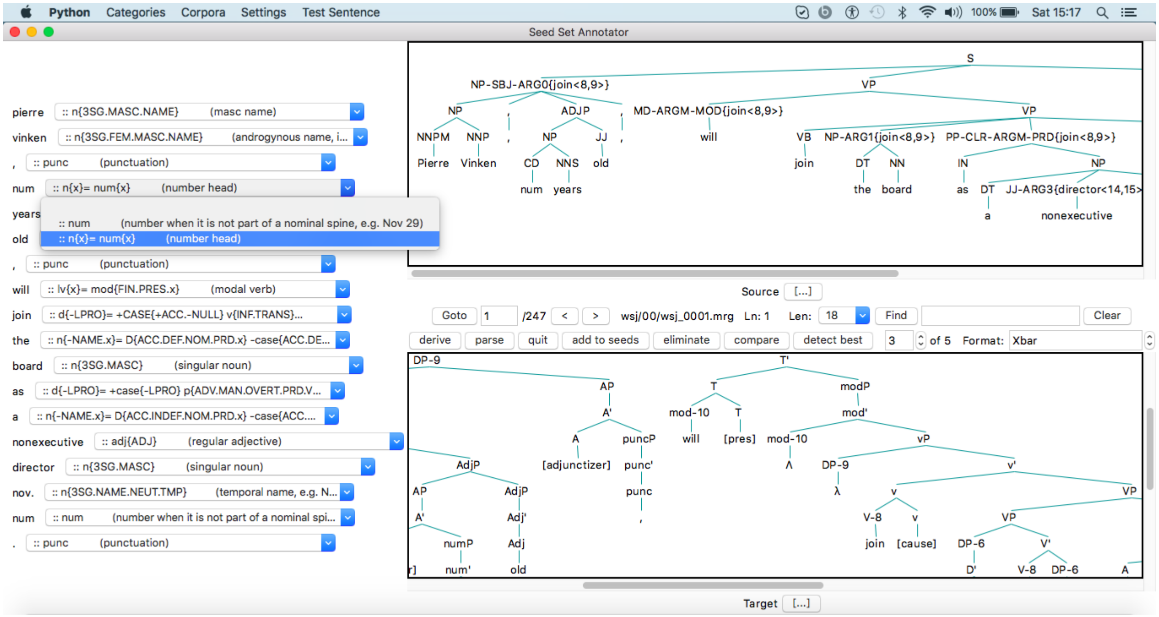


Figure 1: Autobank architecture.



pierre	:: n{3SG.MASC.NAME} (masc name)
vinken	:: n{3SG.FEM.MASC.NAME} (androgynous name, i...
,	:: punc (punctuation)
num	:: n{x}= num{x} (number head)
years	:: num (number when it is not part of a nominal spine, e.g. Nov 29) :: n{x}= num{x} (number head)
old	
,	:: punc (punctuation)
will	:: lv{x}= mod{FIN.PRES.x} (modal verb)
join	:: d{-LPRO}= +CASE{+ACC.-NULL} v{INF.TRANS}...
the	:: n{-NAME.x}= D{ACC.DEF.NOM.PRD.x} -case{ACC.DE...
board	:: n{3SG.MASC} (singular noun)
as	:: d{-LPRO}= +case{-LPRO} p{ADV.MAN.OVERT.PRD.V...
a	:: n{-NAME.x}= D{ACC.INDEF.NOM.PRD.x} -case{ACC....
nonexecutive	:: adj{ADJ} (regular adjective)
director	:: n{3SG.MASC} (singular noun)
nov.	:: n{3SG.NAME.NEUT.TMP} (temporal name, e.g. N...
num	:: num (number when it is not part of a nominal spi...
.	:: punc (punctuation)

Goto 1 /247 wsj/00/wsj_0001.mrg Ln: 1 Len: 18

3 of 5 Format: Xbar

Figure 2: The main annotation environment.

CCG, EDMG is a strongly lexicalized and derivational formalism, with all subcategorization and linear order information encoded on lexical items. EDMG categories are sequences of features ordered from left to right which are checked and deleted as the derivation proceeds. For example, the hypothetical category, d= =d v[6], could be used to represent a transitive verb that first looks to its right for an object with d as its first feature (i.e. a DP), before selecting a DP subject on its left and thus yielding a constituent of category v, i.e. a VP.

MGParse category features can also include subcategorization and agreement properties and requirements, and allow for percolation of such features up the tree via a simple unification mechanism[7]. In lieu of any statistical model, this enables the human annotator to tightly constrain the grammar and thus reduce the amount of local and global ambiguity present during manual and automatic annotation. For example, the category: `v{+TRANS.x}=` +case`{+ACC.-NULL}` =d lv`{TRANS.x}`, could be used for the null causative light verb (so-called *little v*) that is standardly assumed in Minimalism to govern a main transitive verb. It specifies that its VP complement must have the property TRANS and that the ob-

[6] This category (which is actually used for ditransitives by MGParse) is similar to the CCG category (S\NP)/NP.

[7] As in CCG, such unification is limited to atomic property values rather than the sorts of unbounded feature structures found in Head-Driven Phrase Structure Grammars; unification here is therefore not re-entrant.

ject whose case feature it will check (in this case via covert movement) must have the property ACC but must not have the property NULL, and that following selection of an AGENT DP specifier the resulting vP will have the property TRANS. Furthermore, any additional properties carried by the VP complement (such as PRES, 3SG etc.) will be percolated onto the vP (or rather onto its selectee (lv) feature) owing to the *x* variable.

Both overt and null (i.e. phonetically silent) MG categories can be added to the system and later modified using the menu system at the top of the screen. Any time the user attempts to modify a category, the system will first reparse any trees in the seed set containing that category to ensure that the same Xbar tree can still be generated for the sentence in question following the modification[8].

Once the user has selected an MG category for each word in the sentence, clicking `parse` causes MGParse to return all possible parses using these categories. Parsing without annotating some or all of the words in the sentence is also possible: MGParse will simply try all available MG categories already associated with a word's PTB preterminal in the seed set. Once returned, the trees can be viewed in several formats, including multiple MG derivation tree formats, and Xbar tree and MG (bare phrase structure) derived tree formats. Candidate parses can be viewed side-by-side for comparison, and there is the option to perform a diff on the bracketings, and to eliminate incorrect candidates from consideration. Once the user has identified the correct tree, they can click `add to seeds` to save it; seeds can be viewed and removed at any point using the native file system.

There will inevitably be occasions when the parser fails to return any parses. In these cases it is useful to build up the derivation step-by-step to identify the point where it fails, and Autobank provides an interface for doing just this (see fig 3). Whereas in annotation mode null heads were kept hidden for simplicity, in derivation mode the entire null lexicon is available, along with the overt categories the user selected on the main annotation screen. Other features of the system include a test sentence mode, a corpus stats display, a facility for automatically detecting the best candidate MG parse (to test the performance of the automatic annotator, and speed up annotation), a parser settings menu, and an option for backing up all data.

The extreme succinctness of the (strongly) lexicalized (ED)MG formalism, as discussed in Stabler (2013), means that seed set creation can be a relatively rapid process. Like CCGs, MGs have abstract rule schemas which generalize across categories, thus dramatically reducing the size of the grammar. Taking CCGbank's 1300 categories as an approximate upper bound, a researcher working five days a week on the annotation process and adding 20 MG categories per day to the system should have added enough MG categories to parse all the sentences of the PTB within 3 months.

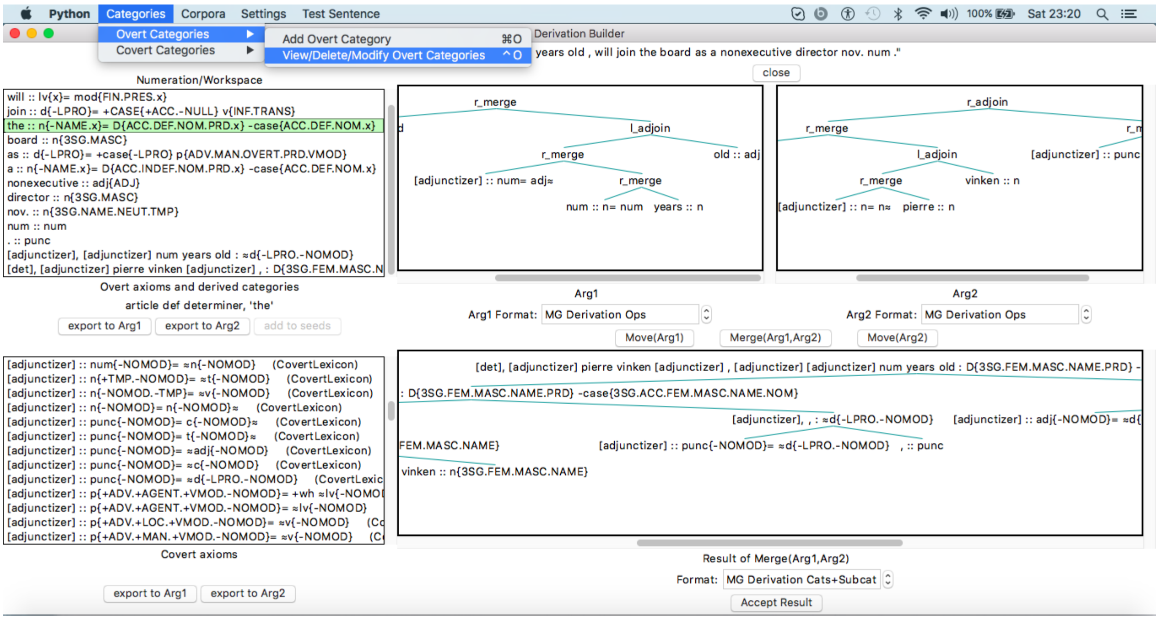
4 The automatic annotation phase

Once the user has created an initial seed set, they can select `auto generate corpus` from the `corpus` menu. This prompts the system to extract a set of dependency mappings and lexical category mappings holding between each seed MG tree and its PTB source tree. To achieve this, the system traverses the PTB and MG trees and extracts a Collins-style dependency tuple for every non-head child of every non-terminal in each tree. The tuples include the head child and non-head child categories, the parent category, any relevant function tags, the directionality of the dependency, and the head child's and non-head child's head words and spans. Where there are multiple tuples in a tree with the same head and non-head word spans, these are grouped together into a *chain*.

For example, for the sentence *the doctor examined Jack*, the AGENT subject NP in a PTB-style tree would yield the following dependency: [VP, *examined*, `<2,3>`, NP, *doctor*, `<1,2>`, S, [ARG0, SUBJ], left]. Many Minimalists assume that AGENT subjects are base-generated inside the verb phrase (Koopman and Sportiche, 1991) in spec-vP, before moving to the surface subject position in spec-TP. Although in its surface position the subject is a dependent of the T(ense) head, it is the lexical verb which is the semantic head of the extended projection (i.e. of the CP clause containing it), and both syntactic and semantic heads are used here when generating the tuples[9]. Two tuples are therefore extracted for the dependency

[8] In general, subcategorization properties and requirements are the only features on an MG category that can be modified without first removing all seed parses containing that category as they do not affect the Xbar tree's geometry (though they can license or prevent its generation).

[9] This also allows the system to capture certain systematic changes in constituency and recognize, for instance, that temporal adjuncts tagged with a TMP label and attaching to VP in the PTB should attach to TP in the MG tree.



Python
Categories
Corpora
Settings
Test Sentence
Sat 23:20
Overt Categories
Covert Categories
Add Overt Category ⌘O
View/Delete/Modify Overt Categories ^O
Derivation Builder
years old , will join the board as a nonexecutive director nov. num ."
close
Numeration/Workspace
r_merge
l_adjoin
r_merge
old :: adj
[adjunctizer] :: num= adj≈
r_merge
num :: n= num
years :: n
r_adjoin
r_merge
l_adjoin
[adjunctizer] :: punc
r_merge
vinken :: n
[adjunctizer] :: n= n≈
pierre :: n
Arg1
Arg2
Move(Arg1)
Merge(Arg1,Arg2)
Move(Arg2)
export to Arg1
export to Arg2
add to seeds
Covert axioms
export to Arg1
export to Arg2
[det], [adjunctizer] pierre vinken [adjunctizer] , [adjunctizer] [adjunctizer] num years old : D{3SG.FEM.MASC.NAME.PRD} -
: D{3SG.FEM.MASC.NAME.PRD} -case{3SG.ACC.FEM.MASC.NAME.NOM}
[adjunctizer], , : ≈d{-LPRO.-NOMOD}
[adjunctizer] :: adj{-NOMOD}= ≈d{
FEM.MASC.NAME}
[adjunctizer] :: punc{-NOMOD}= ≈d{-LPRO.-NOMOD}
, :: punc
vinken :: n{3SG.FEM.MASC.NAME}
Result of Merge(Arg1,Arg2)
Accept Result

```
will :: lv{x}= mod{FIN.PRES.x}
join :: d{-LPRO}= +CASE{+ACC.-NULL} v{INF.TRANS}
the :: n{-NAME.x}= D{ACC.DEF.NOM.PRD.x} -case{ACC.DEF.NOM.x}
board :: n{3SG.MASC}
as :: d{-LPRO}= +case{-LPRO} p{ADV.MAN.OVERT.PRD.VMOD}
a :: n{-NAME.x}= D{ACC.INDEF.NOM.PRD.x} -case{ACC.DEF.NOM.x}
nonexecutive :: adj{ADJ}
director :: n{3SG.MASC}
nov. :: n{3SG.NAME.NEUT.TMP}
num :: num
. :: punc
[adjunctizer], [adjunctizer] num years old : ≈d{-LPRO.-NOMOD}
[det], [adjunctizer] pierre vinken [adjunctizer] , : D{3SG.FEM.MASC.N
```

Overt axioms and derived categories

article def determiner, 'the'

```
[adjunctizer] :: num{-NOMOD}= ≈n{-NOMOD}    (CovertLexicon)
[adjunctizer] :: n{+TMP.-NOMOD}= ≈t{-NOMOD}    (CovertLexicon)
[adjunctizer] :: n{-NOMOD.-TMP}= ≈v{-NOMOD}    (CovertLexicon)
[adjunctizer] :: n{-NOMOD}= n{-NOMOD}≈    (CovertLexicon)
[adjunctizer] :: punc{-NOMOD}= c{-NOMOD}≈    (CovertLexicon)
[adjunctizer] :: punc{-NOMOD}= t{-NOMOD}≈    (CovertLexicon)
[adjunctizer] :: punc{-NOMOD}= ≈adj{-NOMOD}    (CovertLexicon)
[adjunctizer] :: punc{-NOMOD}= ≈c{-NOMOD}    (CovertLexicon)
[adjunctizer] :: punc{-NOMOD}= ≈d{-LPRO.-NOMOD}    (CovertLexic
[adjunctizer] :: p{+ADV.+AGENT.+VMOD.-NOMOD}= +wh ≈lv{-NOMOD
[adjunctizer] :: p{+ADV.+AGENT.+VMOD.-NOMOD}= ≈lv{-NOMOD}
[adjunctizer] :: p{+ADV.+LOC.+VMOD.-NOMOD}= ≈v{-NOMOD}    (Co
[adjunctizer] :: p{+ADV.+MAN.+VMOD.-NOMOD}= ≈v{-NOMOD}    (C
```

Arg1 Format: MG Derivation Ops

Arg2 Format: MG Derivation Ops

Format: MG Derivation Cats+Subcat

Figure 3: The step-by-step derivation builder.

between the verb and the subject and together they form a chain representing the latter's deep and surface positions. The system then establishes mappings from PTB tuples/chains to MG tuples/chains which share the same head and non-head spans (this includes instances where the relation between the head and dependent has been reversed).

Each mapping is stored in three forms of varying degrees of abstraction. In the first, all word-specific information (i.e. the head and non-head words and their spans) is deleted; in the second the spans and non-head word are deleted, but (a lemmatized version of) the head word is retained, while in the third the spans are deleted but both the (lemmatized) head and non-head words are retained. The fully reified mapping for our subject dependency, for instance, would be: [VP, *examine*, NP, *doctor*, S, [ARG0, SUBJ], left] —> [[T', *examine*, DP, *doctor*, TP, left], [v', *examine*, DP, *doctor*, vP, left]]. Including both abstract and reified mappings allows the system to recognise not only general phrase structural correspondences, but also more idiosyncratic mappings conditioned by specific lexical items, as in the case of idioms and light verb constructions, for instance.

The system next parses the remaining sentences, selecting a set of potential MG categories for each word in each sentence using the lexical category mappings[10]. Whenever a dependency mapping is discovered which has previously been seen in the seeds, the MG tree containing it is awarded with a point; the candidate with the most points is added to the Auto MGbank. The abstract mapping above, for instance, ensures that for transitive sentences, trees containing the subject trace in spec vP are preferred. The user can choose to specify the number and maximum string length of the trees that are automatically generated (together with a timeout value for the parser) and can then inspect the results and transfer any good trees into the seed corpus, thereby rapidly expanding it.

5 Conclusion

Autobank is a GUI tool currently being used to semi-automatically construct MGbank, a deep Minimalist version of the PTB. Minimalism is a lively theory, however, and in continual flux. Autobank was therefore designed with reusability in mind, in the hope that other researchers will use it to create alternative Minimalist treebanks, either from scratch or by modifying an existing one, and to stimulate greater interest in computational Minimalism and statistical MG parsing. The system could also potentially be adapted for use with other source and (lexicalised) target formalisms.

[10] Note that there will be many MG categories for every PTB category, which will make parsing quite slow for certain sentences. To ameliorate this, once enough seeds have been added, an MG supertagger (see Lewis and Steedman (2014)) will be trained and used to reduce the amount of lexical ambiguity; to improve things further, multiple sentences will be processed in parallel during automatic annotation.

Acknowledgements

Many thanks to Ed Stabler for providing the initial inspiration for this project, and for his subsequent guidance and encouragement last summer. The work described in this paper was funded by Nuance Communications Inc. and the Engineering and Physical Sciences Research Council. I would also like to thank the reviewers for their comments and also my supervisors Mark Steedman and Shay Cohen for their help and guidance during the development of the Autobank system and the writing of this paper.

References

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2006. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jessica Ficler and Yoav Goldberg. 2016. Coordination annotation extension in the penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, German, Volume 1: Long Papers*, pages 834–842.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1974–1981.

Tim Hunter and Chris Dyer. 2013. Distributions on minimalist grammar derivations. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 1–11, Sofia, Bulgaria, August. The Association of Computational Linguistics.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85(2-3):211–258.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with semi-supervised supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Mitch Marcus, Beatrice Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proceedings of HLT-EACL Workshop: Frontiers in Corpus Annotation*.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Edward Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics (LACL'96)*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, New York. Springer.

Edward Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5:611–633.

Mark Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monograph 30. MIT Press, Cambridge, MA.

John Torr and Edward P. Stabler. 2016. Coordination in minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the Twelfth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+12)*, pages 1–17.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, June. ACL.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D. Hwang, Claire Bonial, Jinho Choi, Aous Mansouri, Maha Foster, Abdel aati Hawwary, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston. 2012. *OntoNotes Release 5.0 with OntoNotes DB Tool v0.999 beta*.