LingPars, a Linguistically Inspired, Language-Independent Machine Learner for Dependency Treebanks

Eckhard Bick

Institute of Language and Communication
University of Southern Denmark
5230 Odense M, Denmark
eckhard.bick@mail.dk

Abstract

This paper presents a Constraint Grammar-inspired machine learner and parser, LingPars, that assigns dependencies to morphologically annotated treebanks in a function-centred way. The system not only bases attachment probabilities for PoS, case, mood, lemma on those features' function probabilities, but also uses topological features like function/PoS n-grams, barrier tags and daughter-sequences. In the CoNLL shared task, performance was below average on attachment scores, but a relatively higher score for function tags/deprels in isolation suggests that the system's strengths were not fully exploited in the current architecture.

1 Introduction

This paper describes LingPars, a Constraint Grammar-inspired language-independent treebank-learner developed from scratch between January 9th and March 9th 2006 in the context of the CoNLL-X 2006 shared task (http://nextens.uvt.nl/~conll/), organized by Sabine Buchholz, Erwin Marsi, Yval Krymolowski and Amit Dubey. Training treebanks and test data were provided for 13 different languages: Arabic (Smrž et al. 2002), Chinese (Chen et al. 2003), Czech (Hajič et al. 2001), Danish (Kromann 2003), Dutch (van der Beek et al. 2002), German (Brants et.al 2002), Japanese (Kawata and Bartels), Portuguese (Afonso et al. 2002), Slovene (Džerosky et al. 2006), Spanish (Palomar et al. 2004), Swedish (Nilsson et al. 2005), Turkish (Oflazer et al. 2003 and Nart et.al 2003), Bulgarian (Simov et al. 2005). A number of these treebanks were not originally annotated in dependency style, but transformed from constituent tree style for the task, and all differ widely in terms of tag granularity (21-302 part-of-speech tags, 7-82 function labels). Also, not all treebanks included morphological information, and only half offered a lemma field. Such descriptive variation proved to be a considerable constraint for our parser design, as will be explained in chapter 2. No external resources and no structural preprocessing were used[1].

2 Language independence versus theory independence

While manual annotation and/or linguistic, rule-based parsers are necessary for the creation of its training data, only a machine learning based parser (as targeted in the CoNNL shared task) can hope to be truly language independent in its design. The question is, however, if this necessarily implies independence of linguistic/descriptive theory.

In our own approach, LingPars, we thus departed from the Constraint Grammar descriptive model (Karlsson et al. 2005), where syntactic function tags (called DEPREL or dependency relations in the shared task) rank higher than dependency/constituency and are established *before* head attachments, rather than vice versa (as would be the case for many probabilistic, chunker based systems, or

[1]The only exception is what we consider a problem in the dependency-version of the German TIGER treebank, where postnominal attributes of nouns appear as dependents of that noun's head if the latter is a preposition, but not otherwise (e.g. if the head's head is a preposition). LingPars failed to learn this somewhat idiosyncratic distinction, but performance improved when the analysis was pre-processed with an additional np-layer (to be re-flattened after parsing.).

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 171–175, New York City, June 2006.

the classical PENN treebank descriptive model). In our hand-written, rule based parsers, dependency treebanks are constructed by using sequential attachment rules, generally attaching functions (e.g. subject, object, postnominal) to forms (finite verb, noun) or lexical tags (tense, auxiliary, transitive), with a direction condition and the possibility of added target, context or barrier conditions (Bick 2005).

In LingPars, we tried to mimic this methodology by trying to learn probabilities for both CG style syntactic-function contexts and function-to-form attachment rules. We could not, however, implement the straightforward idea of learning probabilities and optimal ordering for an existing body of (manual) seeding rules, because the 13 treebanks were not harmonized in their tag sets and descriptive conventions[2].

As an example, imagine a linguistic rule that triggers "subclause-hood" for a verb-headed dependency-node as soon as a subordinator attaches to it, and then, implementing "subclause-hood", tries to attach the verb not to the root, but to another verb left of the subordinator, or right to a root-attaching verb. For the given set of treebanks probabilities and ordering priorities for this rule cannot be learned by one and the same parser, simply because some treebanks attach the verb to the subordinator rather than vice versa, and for verb chains, there is no descriptive consensus as to whether the auxiliary/construction verb (e.g. Spanish) or the main verb (e.g. Swedish) is regarded as head.

3 System architecture

The point of departure for pattern learning in LingPars were the fine-grained part of speech (PoS) tags (POSTAG) and the LEMMA tag. For those languages that did not provide a lemma tag, lowercased word form was used instead. Also, where available from the FEATS field and not already integrated into the PoS tag, the following information was integrated into the PoS tag:

a) *case*, which was regarded as a good predictor for function, as well as a good dependency-indicator for e.g. preposition- and adnominal attachment

b) *mood/finiteness,* in order to predict subordination and verb chaining, especially in the absence of auxiliary class information in the FEATS field

c) *pronoun subclass,* in order to predict adnominal vs. independent function as well as subordinating function (relatives and interrogatives)

A few treebanks did not classify subordinating words as conjunctions, relatives, interrogatives etc., but lumped them into the general adverb and pronoun classes. Danish is a case in point - here, the treebank classified all non-inflecting words as PoS 'U'[3]. Our solution, implemented only for Danish and Swedish, was to introduce a list of structure-words, that would get their PoS appended with an '-S', enabling the learner to distinguish between e.g. "ordinary" ADV, and "structural" ADV-S.

3.1 The parser

In a first round, our parser calculates a preference list of functions and dependencies for each word, examining all possible mother-daughter pairs and n-grams in the sentence (or paragraph). Next, dependencies are adjusted for function, basically summing up the frequency-, distance- and direction-calibrated function→PoS attachment probabilities for all contextually allowed functions for a given word. Finally, dependency probabilities are weighted using linked probabilities for possible mother-, daughter- and sister-tags in a second pass.

The result are 2 arrays, one for possible daughter→mother pairs, one for word:function pairs. Values in both arrays are normalized to the 0..1 interval, meaning that for instance even an originally low probability, long distance attachment will get high values after normalization if there are *few or no* competing alternatives for the word in question.

LingPars then attempts to "effectuate" the dependency (daughter→mother) array, starting with the - in normalized terms - highest value[4]. If the daughter candidate is as yet unattached, and the dependency does not produce circularities or crossing branches, the corresponding part of the (ordered) word:function array is calibrated for the suggested dependency, and the top-ranking function chosen.

In principle, one pass through the dependency array would suffice to parse a sentence. However,

[2] Neither was there time (and for some languages: reading knowledge) to write the necessary converters to and from a normalized standard formalism for each treebank.

[3] For the treebank as such, no information is lost, since it will be recoverable from the function tag. In a training situation, however, there is much less to train on than in a treebank with a more syntactic definition of PoS.

[4] Though we prefer to think of attachments as bottom-up choices, the value-ordered approach is essentially neither bottom-up nor top-down, depending on the language and the salience of relations in a sentence, all runs had a great variation in the order of attachments. A middle-level attachment like case-based preposition-attachment, for instance, can easily outperform (low) article- or (high) top-node-attachment.

due to linguistic constraints like uniqueness principle, barrier tags and "full" heads[5], some words may be left unattached or create conflicts for their heads. In these cases, weights are reduced for the conflicting functions, and increased for all daughter→mother values of the unattached word. The value arrays are then recomputed and rerun. In the case of unattached words, a complete rerun is performed, allowing problematic words to attach before those words that would otherwise have blocked them. In the case of a function (e.g subject uniqueness) conflict, only the words involved in the conflict are rerun. If no conflict-free solution is found after 19 runs, barrier-, uniqueness- and projectivity-constraints are relaxed for a last run[6].

Finally, the daughter-sequence for each head (with the head itself inserted) is checked against the probability of its function sequence (learned not from n-grams proper, but from daughter-sequences in the training corpus). For instance, the constituents of a clause would make up such a sequence and allow to correct a sequence like SUBJ VFIN ARG2 ARG1 into SUBJ VFIN ARG1 ARG2, where ARG1 and ARG2 are object functions with a preferred order (for the language learned) of ARG1 ARG2.

3.2 Learning functions (deprels)

LingPars computes function probabilities (Vf, function value) at three levels: First, each lemma and PoS is assigned *local* (context-free) probabilities for all possible functions. Second, the probability of a given function occurring at a specific place in a function n-gram (func-gram, example (a)) is calculated (with n between 2 and 6). The learner only used *endocentric* func-grams, marking which of the function positions had their head within the func-gram. If no funcgram supported a given function, its probability for the word in question was set to zero. At the third level, for each endocentric n-gram of word classes (PoS), the probability for a given function occurring at a given position in the n-gram (position 2 in example (b)) was computed. Here, only the longest possible n-grams were used by the parser, and first and last positions of the n-gram were used only to provide context, not to assign function probabilities.

(a)>N→2 SUBJ→4 <N→2 AUX MV→4 ACC→5
(b) art→2 n:SUBJ→4 adj→2 v-fin v-inf→4 n→5

3.3 Learning dependencies

In a rule based Constraint Grammar system, dependency would be expressed as attachment of functions to forms (i.e. subject to verb, or modifier to adjective). However, with empty deprel fields, LingPars cannot use functions directly, only their probabilities. Therefore, in a first pass, it computes the probability for the whole possible attachment matrix for a sentence, using learned mother- and daughter-normalized frequencies for attachments of type (a) PoS→PoS, (b) PoS→Lex, (c) Lex→PoS and (d) Lex→Lex, taking into account also the learned directional and distance preferences. Each matrix cell is then filled with a value Vfa ("function attachment value") - the sum of the individual normalized probabilities of all possible functions for that particular daughter given that particular mother multiplied with the preestablished, attachment-independent Vf value for that token-function combination.

Inspired by the BARRIER conditions in CG rule contexts, our learner also records the frequency of those PoS and those functions (deprels) that may appear between a dependent of PoS A and a head of PoS B. The parser then regards all *other,* non-registered interfering PoS or functions as blocking tokens for a given attachment pair, reducing its attachment value by a factor of 1/100.

In a second pass, the attachment matrix is calibrated using the relative probabilities for dependent daughters, dependent sisters and head mother given. This way, probabilities of object and object complement sisters will enhance each other, and given the fact that treebanks differ as to which element of a verb chain arguments attach to, a verbal head can be treated differently depending on whether it has a high probability for another verb (with auxiliary, modal or main verb function) as mother or daughter or not.

Finally, like for functions, n-grams are used to calculate attachment probabilities. For each endocentric PoS n-gram (of length 6 or less), the probabilities of all treebank-supported PoS:function chains and their dependency arcs are learned, and the value for an attachment word pair occurring in the chain will be corrected using both the chain/n-gram probability and the Vf value for the function

[5]Head types with a limited maximum number of dependents (usually, one)
[6]In the rare case of still missing heads or functions, these are computed using probabilities for a simplified set of word classes (mostly the CPOSTAG), or - as a last resort - set to ROOT-attachment.

associated with the dependent in that particular chain. For contextual reasons, arcs central to the n-gram are weighted higher than peripheral arcs.[7]

3.4 Non-projectivity and other language-specific problems

As a general rule, non-projective arcs were only allowed if no other, projective head could be found for a given word. However, linguistic knowledge suggests that non-projective arcs should be particularly likely in connection with verb-chain-dependencies, where subjects attach to the finite verb, but objects to the non-finite verb, which can create crossing arcs in the case of object fronting, chain inversion etc. Since we also noted an error-risk from arguments getting attached to the closest verb in a chain rather than the linguistically correct one[8], we chose to introduce systematic, after-parse raising of certain pre-defined arguments from the auxiliary to the main verb. This feature needs language-dependent parameters, and time constraints only allowed the implementation for Danish, Spanish, Portuguese and Czech. For Dutch, we also discovered word-class-related projectivity-errors, that could be remedied by exempting certain FEATS classes from the parser's general projectivity constraint altogether (*prep-voor* and *V-hulp*)[9].

In order to improve root accuracy, topnode probability was set to zero for verbs with a safe subordinator dependent. However, even those treebanks descriptively supporting this did not all PoS-mark subordinators. Therefore, FEATS-information was used, or as a last resort - for Danish and Swedish - word forms.

A third language-specific error-source was punctuation, because some treebanks (cz, sl, es) allowed punctuation as heads. Also, experiments for the Germanic and Romance languages showed that performance decreased when punctuation was allowed as BARRIER, but increased, when a fine-grained punctuation PoS[10] was included in function and dependency n-grams.

[7]Due to BARRIER constraints, or simply because of insufficient training data in the face of a very detailed tag set, it may be impossible to assign all words n-gram supported functions or dependencies. In the former case, local function probabilities are used, in the latter attachment is computed as function → PoS probability only, using the most likely function.

[8] Single verbs being more frequent than verb chains, the learner tended to generalize close attachment, and even (grand)daughter and (grand)mother conditions could not entirely remedy this problem.

[9]Though desirable, there was no time to implement this for other languages.

[10] Only for Spanish and Swedish was there a subdivision of punctuation PoS, so we had to supply this information in all other cases by adding token-information to the POSTAG field.

4 Evaluation

Because of LingPars' strong focus on function tags, a separate analysis of attachment versus label performance was thought to be of interest. Ill. 1 plots the latter (Y-axis) against the former (X-axis), with dot size symbolizing treebank size. In this evaluation, a fixed training chunk size of 50,000 tokens[11] was used, and tested on a different sample of 5,000 tokens (see also 5/50 evaluation in ill. 2). For most languages, function performance was better than attachment performance (3.2 percentage points on average, as opposed to 0.44 for the CoNLL systems overall), with dots above the hyphenated "diagonal of balance". Interestingly, the graphics also makes it clear that performance was lower for small treebanks, despite the fact that training corpus size had been limited in the experiment, possibly indicating correlated differences in the balance between tag set size and treebank size.

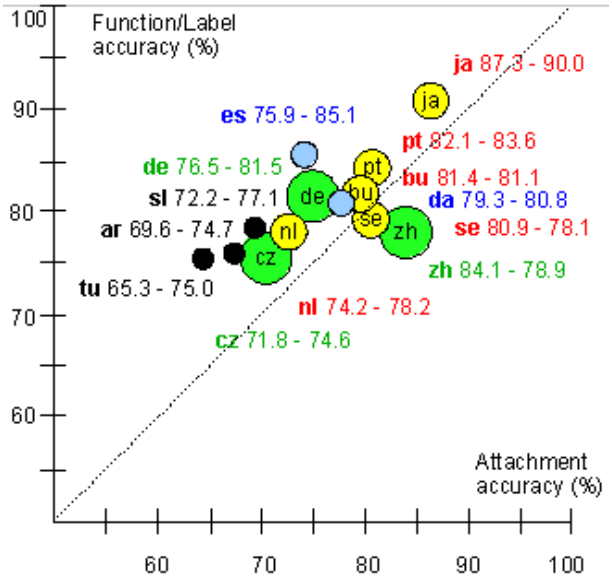

Illustration 1: Attachment accuracy (x-axis) vs. label accuracy (y-axis)

Ill. 2 keeps the information from ill. 1 (5/50-dep and 5/50-func), represented in the two lower lines, but adds performance for maximal training corpus size[12] with (a) a randomly chosen test chunk of 5,000 tokens *not included in the training corpus* (5/all-5) and (b) a 20,000 token chunk *from the training corpus* (20/all). Languages were sorted ac-

[11]Smaller for Slovene and Arabic (for these languages: largest possible)

[12]Due to deadline time constraints, an upper limit of 400,000 lines was forced on the biggest treebanks, when training for unknown test data, meaning that only ½ of the German data and 1/3 of the Czech data could be used.

cording to 20/all-func accuracy. As can be seen from the dips in the remaining (lower) curves, small training corpora (asterisk-marked languages) made it difficult for the parser (1) to match 20/all attachment performance on unknown data, and (2) to learn labels/functions in general (dips in all function curves, even 20/all). For the larger treebanks, the parser performed better (1-3 percentage points) for the full training set than for the 50,000 token training set.

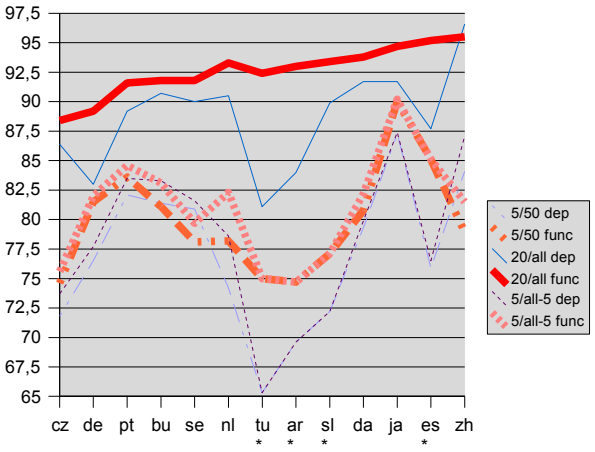

Illustration 2: Performance with different training corpus sizes (upper 2 curves: Test data included)

5 Outlook

We have shown that a probabilistic dependency parser can be built on CG-inspired linguistic principles with a strong focus on function and tag sequences. Given the time constraint and the fact that the learner had to be built from scratch, its performance would encourage further research. In particular, a systematic parameter/performance analysis[13] should be performed for the individual languages. In the long term, a notational harmonization of the treebanks should allow the learner to be seeded with existing hand-written dependency rules.

References

Afonso, S., E. Bick, R. Haber and D. Santos. Floresta Sintá(c)tica: A treebank of Portuguese. In *Proceedings of LREC'02*. pp. 1698-1703 . Paris: ELRA

van der Beek, L. G. Bouma, R. Malouf, G. van Noord. 2002. The Alpino Dependency Treebank. In: *Computational Linguistics in the Netherlands CLIN 2001*. pp. 8-22. Rodopi

Bick, Eckhard. 2005. Turning Constraint Grammar Data into Running Dependency Treebanks. In: Civit, Montserrat & Kübler, Sandra & Martí, Ma. Antònia (ed.), *Proceedings of TLT 2005, Barcelona.* pp.19-2

Brants, S., S. Dipper, S. Hansen, W. Lezius, G. Smith. 2002. The TIGER Treebank. *Proc. of TLT1*, Sozopol

Džerosky, S., T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, A. Žele. 2006. Towards a Slovene Dependency Treebank. In *Proc. of LREC'06*, Genoa

Hajič, J., B. Hladká, and P. Pajas. 2001. The Prague Dependency Treebank: Annotation Structure and Support. In *Proc. of the IRCS Workshop on Linguistic Databases*, pp. 105-114. University of Pennsylvania.

Karlsson, Fred, Atro Vouitilainen, Jukka Heikkilä and A. Anttila. 1995. Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter: Berlin.

Kawata, Y. and J. Bartels. 2000. *Stylebook for the Japanese Treebank in VERBMOBIL*. Universität Tübingen: *Verbmobil-Report 240.*

Chen, Keh-Jiann, Chu-Ren Huang, Feng-Yi Chen, Chi-Ching Luo, Ming-Chung Chang, Chao-Jan Chen, and Zhao-Ming Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In A. Abeille (ed.) *Treebanks Building and Using Parsed Corpora*. Dordrecht:Kluwer, pp231-248.

Kromann, M. T. 2003. The Danish Dependency Treebank. In J. Nivre and E. Hinrichs (ed.) *Proceedings of TLT2003.* Växjö University Press, Sweden

Nart, B. Atalay, Kemal Oflazr, Bilge Say. 2003. The Annotation Process in the Turkish Treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora - LINC 2003.* Budapest

Nilsson, J, J. Hall and J. Nivre. 2005. MAMBA Meets TIGER: Reconstructing a Swedish Treebank from Antiquity. In *Proceedings NODALIDA 2005.* Joenssu

Oflazer, K., B. Say, D.Z. Hakkani-Tür, G. Tür. 2003. Building a Turkish Treebank. In A. Abeillé (ed.) *Building and Exploiting Syntactically-annotated Corpora.* Kluwer

Palomar, M. et. al. 2004. Construcción de una base de datos de árboles sintáctico-semánticos para el catalán, euskera y castellano. In: *Proceedings of SEPLN XX*, pp 81-88. Barcelona: ISSN 1135-5948

Simov, K., P. Osenova, A. Simov, M. Kouylekov. 2004. Design and Implementation of the Bulgarian HPSG-based Treebank. In E. Hinrichs and K. Simov (ed.), *Journal of Research on Language and Computation,* Vol. 2, No. 4 , pp. 495-522. Kluwer

Smrž, Otakar, Jan Šnaidauf, and Petr Zemánek. 2002. Prague Dependency Treebank for Arabic: Multi-Level Annotation of Arabic corpus. In *Proceedings of the International Symposium on Processing of Arabic*, pages 147-155, Manouba, Tunisia, April 2002.

[13]Parameters like uniqueness and directedness are already learned by the system (through probability thresholds), while others, like function weights, structural word classes and frequency thresholds for barriers and lexeme n-grams are used now, but with a fixed value for all languages.