Assigning Deep Lexical Types

João Silva and António Branco

University of Lisbon
Departamento de Informática, Edifício C6
Faculdade de Ciências, Universidade de Lisboa
Campo Grande, 1749-016 Lisboa, Portugal
{jsilva,antonio.branco}@di.fc.ul.pt

Abstract. Deep linguistic grammars provide complex grammatical representations of sentences, capturing, for instance, long-distance dependencies and returning semantic representations, making them suitable for advanced natural language processing. However, they lack robustness in that they do not gracefully handle words missing from the lexicon of the grammar. Several approaches have been taken to handle this problem, one of which consists in pre-annotating the input to the grammar with shallow processing machine-learning tools. This is usually done to speed-up parsing (supertagging) but it can also be used as a way of handling unknown words in the input. These pre-processing tools, however, must be able to cope with the vast tagset required by a deep grammar. We investigate the training and evaluation of several supertaggers for a deep linguistic processing grammar and report on it in this paper.

Keywords: supertagging, deep grammar.

1 Introduction

Parsing is one of the fundamental tasks in Natural Language Processing and a critical for many applications. Many of the most commonly used parsers rely on probabilistic approaches and are obtained by inferring a language model over a dataset of annotated sentences. Though these parsers always produce some analysis of their input sentences, they do not go into deep linguistic analysis.

Deep grammars seek to make explicit highly detailed linguistic phenomena and produce complex grammatical representations for their input sentences. In particular, they are able to capture long-distance dependencies and produce the semantic representation of a sentence. Although there is a great variety of parsing algorithms (see [1] for an overview), they all require a lexical look-up initialization step that, for each word in the input, returns all its possible syntactic categories. From this it follows that if any of the words in a sentence is not present in the lexicon—an *out-of-vocabulary* (OOV) word—a full parse of that sentence is impossible to obtain. Given that novelty is one of the defining characteristics of natural languages, unknown words will eventually occur. Hence, being able to handle OOV words is of paramount importance if one wishes to use a parser to analyze unrestricted texts with an acceptable coverage. Another important issue is that of words that may bear more than one syntactic category. The combinatorial explosion of lexical and syntactic ambiguity may also hinder parsing due to the increased requirements in terms of parsing time and memory usage. Thus, even if there are no OOV words in the input, assigning syntactic categories to words prior to parsing may be desirable for efficiency reasons.

For shallower approaches, like constituency parsing, it suffices determining the part-of-speech (POS), so pre-processing the input with a POS tagger is a common and effective way to tackle either of these problems. However, the information contained in the lexicon of a deep grammar is much more fine-grained, including, in particular, the subcategorization frame of the word, which further constraints what can be taken as a well-formed sentence by imposing several restrictions on co-occurring words. Thus, what for a plain POS tagger corresponds to a single tag is often expanded into hundreds of different distinctions, and hence tags, when at the level of detail required by a deep grammar. For instance, the particular grammar we use for the study reported here has a lexicon with roughly 160 types of verb and 200 types of common noun.[1] While the grammar may proceed with the analysis knowing only the POS category of a word, it does so at the cost of vastly increased ambiguity which may even lead the grammar to accept ungrammatical sentences as valid. This has lead to research that specifically targets annotation with a tagset suitable for deep grammars.

In this paper we investigate several machine-learning approaches to supertagging and compare them with state-of-the-art results. In particular, we experiment with a support vector machine algorithm, a novel approach to supertagging.

2 Related Work

The two main approaches to assigning lexical types for a deep grammar can be divided in terms of how they resolve lexical ambiguity. In lexical acquisition are approaches that try to discover all the types a given unknown word may occur with, effectively creating a new lexical entry. However, at run-time, it is still up to the grammar using the newly acquired entry to choose which of its possible types is the correct one for each particular occurrence of that word. In supertagging are approaches that assign, typically on-the-fly at run-time, a single lexical type to a particular occurrence of a word. Their rationale is not to acquire a new entry to record in the lexicon, but to allow the grammar to keep parsing despite the occurrence of OOV words, or to ease parsing by reducing ambiguity. The approach reported in this paper falls under the umbrella of supertagging.

2.1 Supertagging

POS tagging relies on a small window of context to achieve a limited form of syntactic disambiguation [2]. As such, it is commonly used prior to parsing as a way of reducing ambiguity by restricting words to a certain category, leading to a greatly reduced search space, faster parsing and less demanding memory requirements. Supertagging can be seen as an extension of this idea to a richer tagset, in particular to one that includes information on subcategorization frames.

[1] For instance, in the deep grammar we are using for the study reported here, the lexical type `noun-common-2comps_de-com` is assigned to common nouns that have two complements, the first introduced by "de" and the second by "com".

In [3], a supertagger using a trigram model was applied to a Lexicalized Tree Adjoining Grammar (LTAG), where each lexical item is associated with one or more trees that localize information on dependencies, even long-range ones, by requiring that all and only the dependents be present in the structure [4]. Training data was obtained by taking sentences from the Wall Street Journal and parsing them with XTAG, a wide-coverage LTAG. In addition, in order to reduce data-sparseness, POS tags were used in training instead of words. Evaluation was performed over 100 held-out sentences from the Wall Street Journal. For a tagset of 365 items, this supertagger achieved 68% accuracy. In a later experiment, this is improved to 92% by smoothing model parameters and additional data [5]. The supertagger can also assign the n-best tags, which increases the chances of it assigning the correct supertag at the cost of leaving more unresolved ambiguity. With 3-best tagging, it achieved 97% accuracy.

In [6,7,8], a supertagger is used with a Combinatory Categorial Grammar (CCG), which uses a set of logical combinators to manipulate linguistic constructions. For our purposes here, it matters only that lexical items receive complex tags that describe the constituents they require to create a well-formed construction. The set of 409 categories to be assigned was selected by taking those that occur at least 10 times in sections 02–21 of a CCG automatic annotation of Penn Treebank. Evaluation was performed over section 00, and achieved 92% accuracy. As with [5], assigning multiple tags increases accuracy. However, instead of using a fixed n-best number of tags—which might be to low, or too high, depending on case at hand—the CCG supertagger assigns all tags with a likelihood within a factor β of the best tag. A value for β as small as 0.1 is enough to boost accuracy up to 97% with an average of only 1.4 tags per word.

2.2 Supertagging for HPSG

[9] present an supertagger for the Alpino Dutch grammar based on hidden Markov models. An interesting feature of their approach is that the training data (2 million sentences of newspaper text) is the output of the parser, thus avoiding the need for a hand-annotated dataset. A gold standard was created by having Alpino choose the best parse for a set of 600 sentences. When assigning a single tag (from a tagset with 2,392 tags), the supertagger achieves an accuracy close to 95%. It is unclear to what extent this can be affected by some bias in the disambiguation module, given that the lexical types in the training dataset and in the gold standard are both automatically picked by Alpino.

[10] use a supertagger with the Enju grammar for English. The novelty in their work comes from filtering invalid tag sequences produced by the supertagger before running the parser. In this approach, a CFG approximation of the HPSG is created with the key property that the language it recognizes is a superset of the parsable supertag sequences. Hence, if the CFG is unable to parse a sequence, that sequence can be safely discarded. This approach achieved a labeled precision and recall for predicate-argument relations of 90% and 86%, respectively, over 2,300 sentences with up to 100 words in section 23 of the Penn Treebank.

[11] uses a supertagger with ERG, another grammar for English. Evaluated over 1,798 sentences, it achieves an accuracy of 91% with a tagset of 615 lexical types. Also for ERG, [12] trains two supertaggers over a 158,000 token dataset, one using the

Table 1. Sentence, token and type counts for the various dataset sizes

		tokens			lexical types		
dataset	sent.	all	verbs	nouns	all	verbs	nouns
base	5,422	51,483	6,453 [12.5%]	9,208 [17.9%]	581	129	140
+ Público	10,727	139,330	14,540 [10.4%]	25,301 [18.2%]	626	133	157
+ Wiki	15,108	205,585	20,869 [10.2%]	37,066 [18.0%]	646	139	160
+ Folha	21,217	288,875	29,683 [10.3%]	52,337 [18.1%]	668	140	165

TnT POS tagger [13] and another using the C&C supertagger [6,7,8], and experiments with various tag granularities. For instance, assigning only POS—a tagset with only 13 tags—is the easiest task, with 97% accuracy, while highly granular supertags formed by POS concatenated with selectional restrictions increases the number of tags to 803, with accuracy dropping to 91%.

3 The Grammar and the Dataset

The deep grammar used here is LXGram, an HPSG grammar for Portuguese [14,15]. It supported the annotation of a corpus by providing the set of possible analyses for a sentence (the parse forest), which is then disambiguated by manually picking the correct analysis from among those returned. This ensures that the syntactic and semantic annotation layers are consistent. The corpus is composed mostly by text from the Público newspaper, which was previously annotated with manually verified shallow information, such as POS and lemmas. After running LXGram and disambiguating the result, we were left with 5,422 sentences annotated with all the information provided by LXGram, though, for this paper, we only keep the lexical type assigned to each token. Type distribution is highly skewed. For instance, the 2 most frequent common noun types cover 57% of all the common noun tokens. Such distributions are usually a problem for machine-learning approaches since the number of instances of the rare categories is too small to properly estimate the parameters of the model.

3.1 Dataset Extension

Ahead we will be interested in determining learning curves for the various classifiers under study, but the current dataset is still small. We thus opted for extending it with automatically annotated data obtained by taking additional sentences from Público (4,381), the Portuguese Wikipedia (5,305) and the Folha de São Paulo newspaper (6,109), processing them with a POS tagger, and running them through LXGram. The cumulative sizes are shown in Table 1.

This is made possible because LXGram has a stochastic disambiguation module that chooses the most likely analysis in the parse forest, instead of requiring a manual choice by a human annotator [15]. Manual evaluation of a 50 sentence sample indicates that this module picks the correct reading in 40% of the cases. If this ratio holds, 60% of the sentences will have an analysis that is wrong in some way, though it is not clear how this translates into errors in the assigned lexical types. For instance, when faced with a case of PP-attachment ambiguity, the module may pick the wrong attachment, which gives the wrong analysis though the lexical types assigned to the tokens may be correct.

The lexical types in the corpus are a subset of all types known to the grammar. Table 1 shows the token count for verbs and nouns and, in brackets, their relative frequency. In addition, it provides a breakdown of the number of types for each dataset. As expected, in a true Zipfian way, the dataset must have a major increase in size for the more rare types to appear. For instance, though there nearly a 6-fold difference in tokens between the base and the largest datasets, the latter only contains 11 verb and 25 noun types more. The reader may recall, from Section 1, that the lexicon of LXGram has 160 types of verb and 200 types of common noun. The largest corpus comes close to covering all of these, but there are still a few that, by not occurring in the dataset, cannot be learned. It is worth noting that, for the experiments reported in this paper, and in order to mitigate data-sparseness issues, words were replaced by their lemmas.

4 The Supertaggers

Assigning HPSG lexical types can be envisaged as POS tagging with an unusually detailed tagset. Hence, the most direct way to quickly create a supertagger is to train an off-the-shelf POS tagger over a corpus where tokens are annotated with the HPSG lexical type assigned by the grammar. Though there is a great variety of approaches to POS tagging, they all now achieve roughly the same scores, indicating that a performance ceiling has likely been reached by machine learning approaches to this task.[2] For the current study we opted for the following tools:

TnT is well known for its efficiency and for achieving state-of-the-art results despite having a simple underlying model. It is based on a second-order hidden Markov model extended with linear smoothing of parameters to address data-sparseness issues and suffix analysis for handling unknown words [13].

SVMTool is a tagger based on support-vector machines (SVM). It is extremely flexible in allowing to define which features should be used in the model (e.g. size of word window, number of POS bigrams, etc.) and the tagging strategy (left to right, bidirectional, number of passes, etc). Due to this flexibility, it is described as being a tagger generator [18].

C&C is a package that includes a CCG parser, a computational semantics tool and a supertagger. It is the latter component, the supertagger, that will be used in our study. It is based on the maximum entropy model described in [6]. From the three tools, it is the only one actually described as being a supertagger. It expects, for instance, that the input tokens be already annotated with base POS categories.

[2] This might not be exactly true. Studies have found that, albeit for a simpler task, learning curves still grow as corpora increases exponentially in size into the billion of tokens [16,17]. However, it is doubtful that a manually revised POS tagged corpora of such size could be created to test whether this effect holds for POS tagging.

Table 2. Accuracy scores (%) for various dataset sizes

	all types			verb types			noun types		
dataset	TnT	C&C	SVM	TnT	C&C	SVM	TnT	C&C	SVM
base	88.99	78.65	88.83	90.01	69.72	90.83	87.11	75.46	88.55
+ Público	90.02	80.93	90.30	89.48	74.06	91.08	92.65	85.31	93.20
+ Wiki	90.80	82.45	91.20	89.58	76.18	91.52	93.84	87.98	94.20
+ Folha	91.52	83.60	92.06	90.94	78.03	92.63	94.95	90.48	95.18

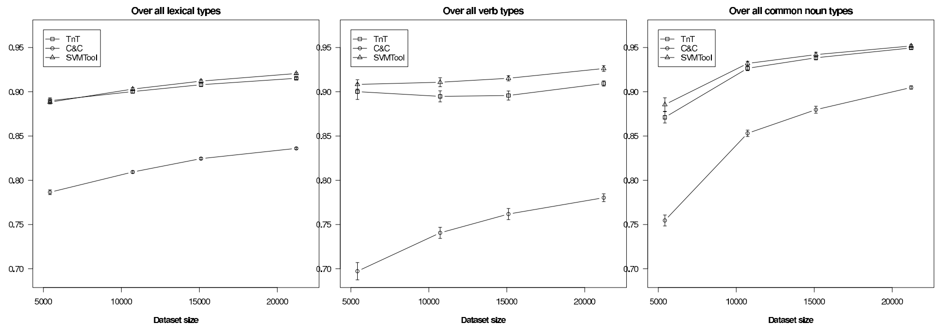

Fig. 1. Learning curves for all types, verb types and noun types

All were run using default parameters. SVMTool used the simplest setting, "M0 LR".[3] The results discussed in the following Section can thus be seen as a baseline for these tools over this task.

5 Evaluation

Evaluation was performed following a 10-fold cross-validation approach over a random shuffle of the sentences in the corpus. Table 2 shows the results obtained over the base dataset of 5,422 sentences. TnT and SVMTool have very similar performance scores, with non-significant differences in accuracy (cf. Fig. 1 for confidence intervals). C&C shows worse performance even though it relies on a more complex model and makes use of the POS tags that are already in the text.[4] Though this might at first seem surprising, it is in line with [12], where C&C also performed worse than TnT. The best results are also close to those of [11] and [12], who got similar overall accuracy scores in their experiments.

To assess the impact of extra training data, we turn to the automatically extended datasets described in Section 3.1. The accuracy scores are summarized in Table 2 while Fig. 1 shows the corresponding plots with the addition of errors bars that represent a 95% confidence interval.

[3] Model 0, left-to-right tagging direction. See [19] for an explanation of these settings.

[4] Recall that the datasets were pre-annotated with POS tags.

There is an improvement in the performance of all taggers as the dataset increases in size. The C&C supertagger was seen to have worse performance over the base dataset. This might be due to it using a more complex model, which needs additional data to properly estimate the parameters, and extra training data would allow it to close the gap to the other tools. The learning curves, however, indicate that this is not the case since the performance of C&C does not seem to be increasing at a fast enough rate for that to happen. TnT and SVMTool show similar scores as the corpus increases in size. Upon reaching the largest dataset, SVMTool pushes ahead with an advantage that, though only of 0.5% points, is likely significant for such a dataset size. The conclusion that can be taken from the remaining tests is similar to what was seen when evaluating over all lexical types—C&C is the worst supertagger, while TnT and SVMTool have similar performance scores. The curves over verb types are quite flat, C&C being the only one that shows any significant improvement, though it might be that, due to its low scores, it still has much room for increasing its accuracy. Since accuracy over verbs does not improve much, the raising curve that is seen over all lexical types is due to an increase in accuracy over other categories, such as common nouns, which show a much more marked improvement.

Each tool has different strengths. SVMTool is better than TnT at annotating verbs, having higher accuracy when tagging this category from early on. This is not so with common nouns, where TnT actually closes up the initial gap to SVMTool and both end up having an indistinguishable score. The best supertagger, SVMTool, shows very good results. Despite having ran using the simplest model, it has better performance than the other supertaggers.

6 Final Remarks and Future Work

In this paper we report on experiments where three supertaggers for a Portuguese HPSG grammar were induced over differently sized datasets. Over the larger dataset, the best supertaggers showed state-of-the-art accuracy of 91–93%, similar to that obtained in related work for English, in particular [11] and [12]. SVM technology had yet to be applied to supertagging. The experiments reported here show that, like in other NLP areas where it was used, it improves over existing techniques. Given how SVM easily incorporates many features, and the flexibility of SVMTool, future work will test how features and tagging strategy can be adjusted to improve performance. For instance, despite having better performance, the configuration used seems poorly suited for an effective annotation of verbs. This makes developing features specifically suited for discriminating the various verb types a promising avenue of research.

References

1. Mitkov, R. (ed.): The Oxford Handbook of Computational Linguistics. Oxford University Press (2004)
2. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing, 1st edn. MIT Press (1999)

3. Bangalore, S., Joshi, A.: Disambiguation of super parts of speech (or supertags): Almost parsing. In: Proceedings of the 15th Conference on Computational Linguistics (COLING), pp. 154–160 (1994)
4. Joshi, A., Schabes, Y.: Handbook of Formal Languages and Automata. In: Tree-Adjoining Grammars. Springer (1996)
5. Bangalore, S., Joshi, A.: Supertagging: An approach to almost parsing. Computational Linguistics 25(2), 237–265 (1999)
6. Clark, S., Curran, J.: Log-linear models for wide-coverage CCG parsing. In: Proceedings of the 8th Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 97–104 (2003)
7. Clark, S., Curran, J.: The importance of supertagging for wide-coverage CCG parsing. In: Proceedings of the 20th Conference on Computational Linguistics (COLING), pp. 282–288 (2004)
8. Clark, S., Curran, J.: Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics 33, 493–552 (2007)
9. Prins, R., van Noord, G.: Reinforcing parser preferences through tagging. Traitment Automatique des Langues 44, 121–139 (2003)
10. Matsuzaki, T., Miyao, Y., Tsujii, J.: Efficient HPSG parsing with supertagging and CFG-filtering. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1671–1676 (2007)
11. Blunsom, P.: Structured Classification for Multilingual Natural Language Processing. Ph.D. thesis, University of Melbourne (2007)
12. Dridan, R.: Using Lexical Statistics to Improve HPSG Parsing. Ph.D. thesis, University of Saarland (2009)
13. Brants, T.: TnT — a statistical part-of-speech tagger. In: Proceedings of the 6th Applied Natural Language Processing Conference and the 1st North American Chapter of the Association for Computational Linguistics, pp. 224–231 (2000)
14. Branco, A., Costa, F.: A computational grammar for deep linguistic processing of Portuguese: LX-Gram, version A.4.1. Technical Report DI-FCUL-TR-08-17, University of Lisbon (2008)
15. Costa, F., Branco, A.: LXGram: A Deep Linguistic Processing Grammar for Portuguese. In: Pardo, T.A.S., Branco, A., Klautau, A., Vieira, R., de Lima, V.L.S. (eds.) PROPOR 2010. LNCS (LNAI), vol. 6001, pp. 86–89. Springer, Heidelberg (2010)
16. Banko, M., Brill, E.: Mitigating the paucity of data problem: Exploring the effect of training corpus size on classifier performance for NLP. In: Proceedings of the 1st Human Language Technology (HLT) Conference (2001)
17. Banko, M., Brill, E.: Scaling to very very large corpora for natural language disambiguation. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and 10th Conference of the European Chapter of the Association for Computational Linguistics, pp. 26–33 (2001)
18. Giménez, J., Màrquez, L.: SVMTool: A general POS tagger generator based on support vector machines. In: Proceedings of the 4th Language Resources and Evaluation Conference (LREC) (2004)
19. Giménez, J., Màrquez, L.: SVMTool: Technical Manual v1.3. TALP Research Center, LSI Department, Universitat Politecnica de Catalunya (2006)