Exploring Syntactic Representations for Native Language Identification

Ben Swanson
Brown University
Providence, RI
chonger@cs.brown.edu

Abstract

Tree Substitution Grammar rules form a large and expressive class of features capable of representing syntactic and lexical patterns that provide evidence of an author's native language. However, this class of features can be applied to any general constituent based model of grammar and previous work has done little to explore these options, relying primarily on the common Penn Treebank annotation standard. In this work we contrast the performance of syntactic features for Native Language Indentification using five different formalisms. The use of different formalisms captures complementary information from second language data, and can be used in combination to yield classification performance superior to any formalism taken on its own.

1 Introduction

Native Language Identification, the automatic determination of an author's native language (L1) from their writing in a second language (L2), follows a general trend of supervised classification using features extracted from text. These systems can be optimized by both classification algorithm selection and the integration of diverse feature sets, and in this work we focus on the latter.

Syntactic features have been shown to provide a strong discriminative signal of an author's native language (Wong and Dras, 2011; Swanson and Charniak, 2012), but little work has been done to explore the various options for representation of syntax of learner text. Many such representations exist, and are routinely employed to improve performance on the widely studied task of parsing the Penn Treebank. Furthermore, most techniques that prove widely successful at this task have publicly available implementations, making them very feasible options for NLI systems.

In this work we investigate the use of Tree Substitution Grammars as features for NLI, focusing on the implication of syntactic paradigm (constituent vs dependency grammar) and the addition of annotations that have proved useful in statistical parsing. A Tree Substitution Grammar (TSG) is an intuitive extension of the Context Free Grammar (CFG) that allows rewrite rules of arbitrary tree structure. Alternatively, a CFG can be seen as a TSG in which the rewrite rules obey the constraint that each is a tree structure of unit depth.

While a collection of parsed data can be potentially generated by a TSG that is exponential in the length of the text, recent techniques allow for the efficient induction of compact grammars (Cohn and Blunsom, 2010). At a high level, this technique employs the rich-get-richer dynamics of a Dirichlet Process to sample derivations for the trees in the training corpus: the more that a rule is used in other derivations, the more likely it is that we will choose it when sampling a derivation.

We follow previous work in stylometry with TSGs for the NLI in that we parse the entirety of the training data and use it to induce a compact TSG using the method described above.[1] We then use the

[1]An alternative method of note that we do not consider in this work is to induce TSG rules on hand-annotated data such as the Penn Treebank, as in Bergsma et al. (2012).

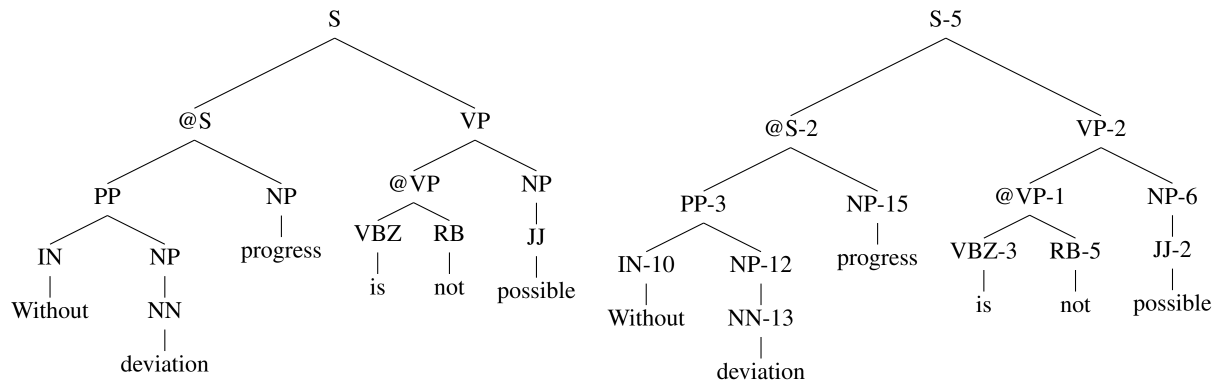


Figure 1: Sample parse trees produced by the Berkeley Parser. An example of what the tree might look like with split symbol annotations is shown on the right.

TSG rules as binary features for supervised classification such that the feature for a TSG rule is triggered on a document if that rule appears in the parse of some derivation of any of its sentences. This description purposefully treats the parsing of text as a black box whose input is plain text and whose output is any valid tree structure. Our work considers five alternatives for this black box, and evaluates the effect of this choice on the NLI Shared Task at the BEA Workshop of NAACL 2013 (Tetreault et al., 2013).

2 Syntactic Representations

We investigate five variations on the output of the parsing process. All five are easily produced by freely available Java software; two with the Berkeley Parser, two with the Stanford Parser, and one with a combination of both software packages.

2.1 Berkeley Constituent Parses

Our first representation reproduces previous work by using the output of the Berkeley Parser (Petrov et al., 2006), one of highest performing systems on the benchmark Penn Treebank task. The basic motivating principle involved is that the traditional nonterminal symbols used in Penn Treebank parsing are too coarse to satisfy the context free assumption of a CFG. To combat this, hierarchical latent annotations are induced that split a symbol into several subtypes, and a larger CFG is estimated on this set of split nonterminals. A sentence is parsed using this large CFG and each resulting symbol is mapped back to its original unsplit supertype to produce the final parse.

One important subtlety of the Berkeley Parser is its default binarization, which we leave intact in our downstream use of its parses. While binarization is normally motivated by the desired cubic complexity of parsing algorithms, it also benefits syntactic stylometry. Consider the nugget of wisdom from the great Frank Zappa shown on the left in Figure 1, in which artificially introduced binarization nodes are marked with the @ symbol.

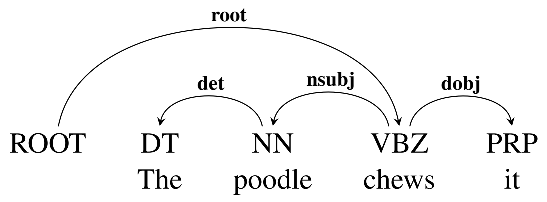
The use of binarization allows us to capture patterns such as verb phrases that begin with ”is not” independent of the following child constituents. The capabilities of TSG rules makes the use of binarization even more apt, as we can easily choose to recover the unbinarized pattern with a slightly larger fragment. This choice will be made in TSG induction based on the frequency with which the combination occurs, which intuitively aligns with our goal of choosing representative features.

The second form that we investigate is identical to the normal Berkeley Parser output, but with the split annotations used in parsing left intact, as shown the right of Figure 1. This parsed sentence shows how each nonterminal is annotated with a split category, and illustrates the potential advantages that this method affords. For example, consider the @VP node in the left-hand tree, whose subtree is generated with a CFG by first choosing to produce a VBZ and RB, and then by lexicalizing each independently. These two lexicalizations are not in fact independent, as can be seen by the combination of ”is” with the RB ”may”, which is impossible al-though each are independently quite likely. Splitting the symbols as shown on the right allows us to create a special RB node that is most likely to produce ”not” and VBZ node likely to produce ”is”. Their likely co-occurrence can then be modeled as shown by a rule with both specialized tags as children.

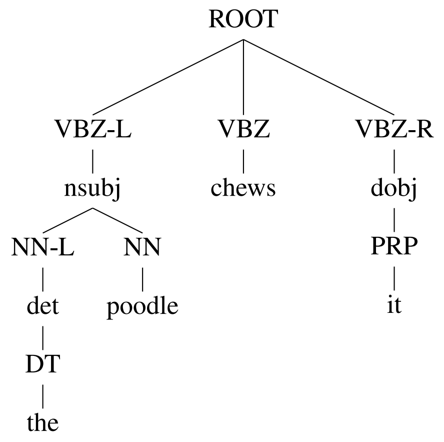
It is worth noting that this particular ability of split symbol grammars to coordinate lexical items is easily captured with the TSG rules that we induce on these parses, regardless of the presence of split symbols. The more orthogonal quality of these split grammars is their ability to categorize symbols that appear in similar syntactic situations. Consider that some adjectives are more likely to appear in ”X is Y” sentences in the ”Y” position, while some are more likely to be used directly to the left of nouns. A split symbol grammar handily captures this trait with a split POS tag, while a TSG cannot associate patterns containing different lexical items on its own.

2.2 Stanford Dependency Parses

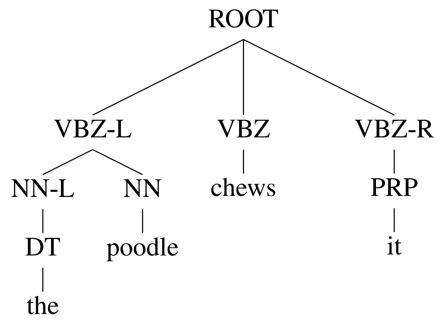
The third and fourth syntactic models we employ are derived from dependency parses produced by the Stanford parser(Marneffe et al., 2006). In its standard form, a dependency parse is a directed tree in which each word except the special ROOT node has exactly one incoming edge and zero to many outgoing edges, where edges represent syntactic dependence. Arcs are labeled with the type of syntactic dependence that they indicate. Following convention, we represent each word in combination with its part of speech tag, as shown in the following example dependency parse.



In order to apply the techniques of TSG induction to dependency parsed data, we implement a conversion from dependency tree to constituent form. The mechanics of this conversion are simple and illustrated in full by the following conversion of the dependency tree shown above, and are similar to transforms used in previous work in unsupervised dependency parsing(Carroll and Charniak, 1992).



Note that it is always the case that the arc labels from the dependency parses are always produced by unary rules. This allows the simple removal of the nodes corresponding to arc labels, yielding our fourth syntactic model.

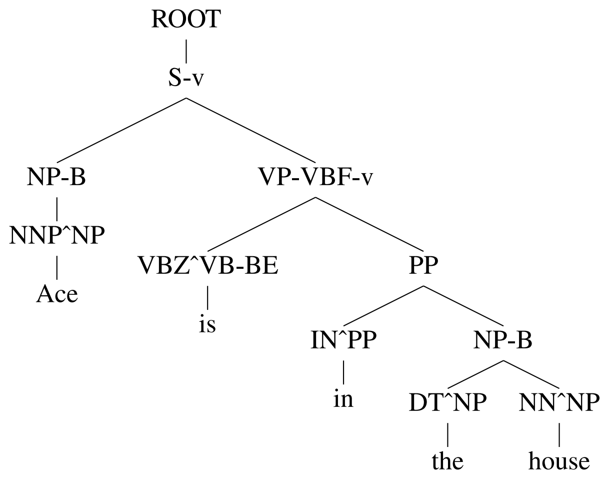


Those familiar with the Stanford Parser may be concerned that the dependency parses used here are determined by a deterministic transform of a constituent parse of Penn Treebank style, and then simply transformed back into constituent form. This is especially concerning when considering the second form in which arc labels have been removed; this form can be constructed directly from the Berkeley Parse form used above, and contains no additional information. Our motivation in the investigation of dependency parses is not that they offer new information, but that they are organized differently than constituent parses. When inducing a TSG, our ability to find a useful connections is impeded by physical distance between structures. In particular, in a dependency parse, the head of the subject and the verb are always contained in some TSG fragment made up of small number of CFG rules, five or four depending on the presence of arc labels. In constituent parses, the presence of modifying phrases can arbitrarily increase this distance.

2.3 Stanford Heuristic Annotations

Our final variation uses the annotations internal to the Stanford Penn Treebank parser, as presented in Klein and Manning (2003). These annotations are motivated in the same way as Berkeley Parser split states, but are deterministically applied to parse trees using linguistic motivations. Besides handling explicit tracking of binarization and parent annotation, several additional annotations are applied, such as the splitting of certain POS tags into useful categories and annotation of some nodes with their number of children or siblings.

For ease of implementation, we do not use the Stanford Parser itself to produce our trees, instead we used our results from the Berkeley Parser. The Stanford Parser annotations were then applied to these trees after binarization symbols were first collapsed. The following tree is an example of the actual annotations applied by this process, and includes a fair subset of the many annotation types that are used. The original symbol in each case is the leftmost string of capital letters in the resulting symbol strings shown.



3 Experiments

We contrast the syntactic formalisms on the NLI shared task experimental setup for the NAACL 2013 BEA workshop. This new data set (Blanchard et al., 2013) consists of TOEFL essays drawn from speakers of 11 different L1 backgrounds. 9900 Essays were supplied as a training set, with an additional 1100 development set essays and 1100 test essays.

Previous work in NLI has relied heavily on the International Corpus of Learner English, but due to significant topic biases along L1 lines in this data set the explicit use of word tokens was frequently limited to a predetermined set of stopwords. With this in mind, the data set for the shared task was balanced across TOEFL essay prompts and proficiency levels. The result was that the participants in this task were not forced to limit the word tokens explicitly employed, with the hopes that mitigating factors had been minimized.

We prepared the data in the five forms described above and induced TSGs on each version of the parsed training set with the blocked sampling algorithm of Cohn and Blunsom (2010). The resulting rules were used as binary feature functions over documents indicating the presence of the rule in some derivation of sentence in that document. We used the Mallet implementation of a log-linear (MaxEnt) classifier with a zero mean Gaussian prior with variance .1 on the classifier's weights. Our results on the development set are shown in Figure 3.

While a range of performance is achieved, when we construct a classifier that simply averages the predictive distributions of all five methods we get better accuracy than any model on its own. We observed further evidence of the orthogonality of these methods by looking at pairs of formalisms and observing how many development set items were predicted correctly by one formalism and incorrectly by another. This was routinely around 10 percent of the development set in each direction for a given pair, implying that gains of up to at least 20 percent classification accuracy are possible with an expert system that approaches oracle selection of which formalism to use.

As our submission to the shared task, we used the Berkeley Parser output in isolation, the average of the five classifiers, and the weighted average of the classifiers using the optimal weights on the development set. The former two models use the development set as additional training data, which is one possible explanation of the slightly higher performance of the equally weighted average model. An-

	ARA	CHI	FRE	GER	HIN	ITA	JPN	KOR	SPA	TEL	TUR	P	R	F
ARA	76	2	4	1	2	2	2	1	4	3	3	76.8	76.0	76.4
CHI	2	86	0	1	1	0	4	4	1	0	1	81.1	86.0	83.5
FRE	2	1	77	3	2	6	2	1	5	1	0	82.8	77.0	79.8
GER	0	1	1	91	1	1	0	0	2	0	3	86.7	91.0	88.8
HIN	2	2	1	2	71	0	0	0	0	20	2	73.2	71.0	72.1
ITA	2	0	2	1	1	84	0	1	7	0	2	79.2	84.0	81.6
JPN	3	4	0	1	0	0	83	7	1	0	1	74.1	83.0	78.3
KOR	1	6	1	1	1	0	20	65	2	1	2	69.1	65.0	67.0
SPA	4	2	4	3	2	12	0	3	66	0	4	71.7	66.0	68.8
TEL	1	2	0	0	16	0	0	0	0	81	0	76.4	81.0	78.6
TUR	6	0	3	1	0	1	1	12	4	0	72	80.0	72.0	75.8

Figure 2: Confusion Matrix and per class results on the final test set evaluation using the evenly averaged model.

other explanation of note is that while the weight optimization was carried out with EM over the likelihood of the development set labels, this did not in correlate positively with classification accuracy; even as we optimized on the development set the accuracy in absolute classification of these items decreased slightly.

The confusion matrix for the evenly averaged model, our best performing system, is shown in Figure 2. The most frequently confused L1 pairs were Hindi and Telegu, Japanese and Korean, and Spanish and Italian. The similarity between Hindi and Telegu is particularly troubling, as they come from two completely different language families and their most obvious similarity is that they are both spoken primarily in India. This suggests that even though the TOEFL corpus has been balanced by topic that there is a strong geographical signal that is correlated with but not caused by native language.

	BP	BPS	DP	DPA	KM	AVG
Acc	74.5	69.3	72.4	73.5	73.5	77.3

Figure 3: The resulting classification accuracies on the development set for the various syntactic forms that we considered. The forms used are plain Berkeley Parses (BP), Berkeley Parses with split symbols (BPS), dependency parses (DP), dependency parses without arc labels (DPA), and the heuristic annotations from (Klein and Manning, 2003) (KM). When the predictive distributions of the five models are averaged (AVG), a higher accuracy is achieved.

	BP	AVG	AVG-EM
Acc	74.7	77.5	77.0

Figure 4: The classification accuracies obtained on the test data using the Berkeley parser output alone (BP), the arithmetic mean of all five predictive distributions (AVG) and the weighted mean using the optimal weights from the development set as determined with EM (AVG-EM)

4 Conclusion

In this work we open investigation of a generally unconsidered variable in syntactic stylometry: the actual syntactic formalism. We examine five potential candidates of which only one has been previously presented in the context of TSG features for NLI. These five formalisms cover both constituent and dependency grammars, and explore the possibility of split state annotations for constituent grammars and the inclusion of arc labels for dependency grammars. We find that the use of different grammar formalisms captures orthogonal information about an author's native language. Furthermore, the combination of different formalisms can be used to increase classification accuracy.

While our results are intriguing, they primarily serve as a proof of concept that syntactic stylometry can benefit from a range of representations and should not be taken as an exhaustive search for the best representations to use. Other syntactic forms exist, and even in our methods there are additional variables that can be adjusted.

One such variable is the number of splits used in the Berkeley Parser when split states are included; the default number that we use in this work is 6, the optimal value for the parsing task, but this may be suboptimal as a representation for feature extraction. Binarization is another easily adjusted variable, with several available options in the literature. For example, binarization can be done that is aware of head attachment. Another option is to binarize more heavily, increasing the ability of TSG fragments to separate sister nodes and find frequent patterns.

Alternative syntactic forms not explored in this work are also available. These include well studied grammars such as Hierarchical Phrase Structure Grammars and Combinatory Categorial Grammars, and transforms that rearrange the tree such as the Left Corner Transform used in Roark and Johnson (1999). Furthermore, the use of the TSG as a feature extractor itself has the potential for extension to more powerful systems such as Tree Adjoining Grammars or Tree Insertion Grammars.

References

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric Analysis of Scientific Articles. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 327–337, Montréal, Canada, June. Association for Computational Linguistics.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. Toefl11: A corpus of non-native english. Technical report, Educational Testing Service.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical Report CS-92-16, Brown University, Providence, RI, USA.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in bayesian tree substitution grammars. In *ACL (Short Papers)*, pages 225–230.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In Proc. Intl Conf. on Language Resources and Evaluation (LREC*, pages 449–454.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Brian Roark and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *ACL*.

Benjamin Swanson and Eugene Charniak. 2012. Native Language Detection with Tree Substitution Grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 193–197, Jeju Island, Korea, July. Association for Computational Linguistics.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, GA, USA, June. Association for Computational Linguistics.

Sze-Meng Jojo Wong and Mark Dras. 2011. Exploiting Parse Structures for Native Language Identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1600–1610, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.