Judging Grammaticality with Count-Induced Tree Substitution Grammars

Francis Ferraro, Matt Post and **Benjamin Van Durme**
Department of Computer Science, and HLTCOE
Johns Hopkins University
{ferraro,post,vandurme}@cs.jhu.edu

Abstract

Prior work has shown the utility of syntactic tree fragments as features in judging the grammaticality of text. To date such fragments have been extracted from derivations of Bayesian-induced Tree Substitution Grammars (TSGs). Evaluating on discriminative coarse and fine grammaticality classification tasks, we show that a simple, deterministic, count-based approach to fragment identification performs on par with the more complicated grammars of Post (2011). This represents a significant reduction in complexity for those interested in the use of such fragments in the development of systems for the educational domain.

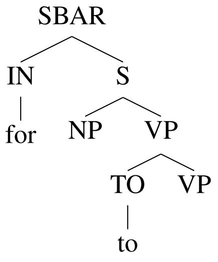
1 Introduction

Automatically judging grammaticality is an important component in computer-assisted education, with potential applications including large-scale essay grading and helping to interactively improve the writing of both native and L2 speakers. While n-gram models have been productive throughout natural language processing (NLP), they are obviously insufficient as models of languages, since they do not model language structure or correspondences beyond the narrow Markov context.

Context-free grammars (CFGs) address many of the problems inherent in n-grams, and are therefore intuitively much better suited for grammaticality judgments. Unfortunately, CFGs used in practice are permissive (Och et al., 2004) and make unrealistic independence and structural assumptions, resulting in "leaky" grammars that overgenerate and thus serve poorly as models of language. However, approaches that make use of the CFG productions as discriminative features have performed better. Cherry and Quirk (2008) improved upon an n-gram baseline in grammatical classification by adjusting CFG production weights with a latent SVM, while others have found it useful to use comparisons between scores of different parsers (Wagner et al., 2009) or the use of CFG productions in linear classification settings (Wong and Dras, 2010) in classifying sentences in different grammaticality settings.

Another successful approach in grammaticality tasks has been the use of grammars with an extended domain of locality. Post (2011) demonstrated that larger syntactic patterns obtained from Tree Substitution Grammars (Joshi, 1985) outperformed the Cherry and Quirk models. The intuitions underlying their approach were that larger fragments are more natural atomic units in modeling grammatical text, and that larger fragments reduce the independence assumptions of context-free generative models since there are fewer substitution points in a derivation. Their grammars were learned in a Bayesian setting with Dirichlet Process priors, which have simple formal specifications (c.f., Goldwater et al. (2009, Appendix A)), but which can become quite complicated in implementation.

In this paper, we observe that fragments used for classification do not require an underlying probabilistic model. Here, we present a simple extraction method that elicits a classic formal non-probabilistic grammar from training data by deterministically counting fragments. Whereas Post parses with his TSG and extracts the Viterbi derivation, we use an



(a) A TSG fragment.

SBAR → IN S
IN → for
S → NP VP
VP → TO VP
TO → to

(b) Equivalent CFG rules.

Figure 1: Equivalent TSG fragment and CFG rules.

off-the-shelf parser and pattern match the fragments in our grammar against the tree. With enough positive and negative training data (in the form of automatic parses of good and bad sentences), we can construct classifiers that learn which fragments correlate with grammaticality. The resulting model results in similar classification accuracy while doing away with the complexity of Bayesian techniques.

2 Tree Substitution Grammars (TSGs)

Though CFGs and TSGs are weakly equivalent, TSGs permit nonterminals to rewrite as tree fragments of arbitrary size, whereas CFG rewrites are limited to depth-one productions. Figure 1 depicts an example TSG fragment and equivalent CFG rules; note that the entire internal structure of 1a is described within a single rewrite.

Unfortunately, learning probabilistic TSGs is not straight-forward, in large part because TSG-specific resources (e.g., large scale TSG-annotated treebanks) do not exist. Approaches to this problem began by taking *all* fragments $\mathcal{F}_{\text{all}}$ in a treebank (Bod, 1993; Goodman, 1996), which resulted in very large grammars composed mostly of fragments very unlikely to generalize.[1] A range of heuristic solutions reduced these grammar sizes to a much smaller, more compact subset of all fragments (Zollmann and Sima'an, 2005; Zuidema, 2007). More recently, more principled models have been proposed, taking the form of inference in Bayesian non-parametric models (Post and Gildea, 2009; Cohn et al., 2009). In addition to providing a formal model for TSGs, these techniques address the overfitting problem of all fragments grammars with priors that discourage large fragments unless there is enough evidence to warrant their inclusion in the grammar. The problem with such approaches, however, is that the sampling procedures used to infer them can be complex, difficult to code, and slow to converge. Although more general techniques have been proposed to better explore the search space (Cohn and Blunsom, 2010; Cohn et al., 2010; Liang et al., 2010), the complexity and non-determinism of these samplers remain, and there are no publicly available implementations.

The underlying premise behind these grammar learning approaches was the need for a probabilistic grammar for parsing. Post (2011) showed that the fragments extracted from derivations obtained by parsing with probabilistic TSGs were useful as features in two coarse-grained grammaticality tasks. In such a setting, fragments are needed for classification, but it is not clear that they need to be obtained from derivations produced by parsing with probabilistic TSGs. In the next section, we describe a simple, deterministic, count-based approach to learning an unweighted TSG. We will then demonstrate (§4) the effectiveness of these grammars for grammaticality classification when fragments are pattern-matched against parse trees obtained from a state-of-the-art parser.

3 Counting Common Subtrees

Rather than derive probabilistic TSGs, we employ a simple, iterative and deterministic (up to tie-breaking) alternative to TSG extraction. Our method extracts $\mathcal{F}_{\langle R,K\rangle}$, the K most common subtrees of size at most R. Though selecting the top K-most-frequent fragments from *all* fragments is computationally challenging through brute force methods, note that if $F \in \mathcal{F}_{\langle R,K\rangle}$, then all subtrees F' of F must also be in $\mathcal{F}_{\langle R,K\rangle}$.[2] Thus, we may incrementally build $\mathcal{F}_{\langle R,K\rangle}$ in the following manner: given r, for $1 \leq r \leq R$, maintain a ranking S, by frequency, of all fragments of size r; the key point is that S may be built from $\mathcal{F}_{\langle r-1,K\rangle}$. Once all fragments of size r have been considered, retain only the top K fragments of the ranked set $\mathcal{F}_{\langle r,K\rangle} = \mathcal{F}_{\langle r-1,K\rangle} \cup S$.[3]

[1] The n-gram analog would be something like storing all 30-grams seen in a corpus.

[2] Analogously, if an n-gram appears K times, then all constituent m-grams, $m < n$, must also appear at least K times.

[3] We found that, at the thresholding stage, ties may be arbitrarily broken with neglible-to-no effect on results.

Algorithm 1 EXTRACTFRAGMENTS (R, K)

```
Assume: Access to a treebank
$S \leftarrow \emptyset$
$\mathcal{F}_{\langle 1,K \rangle} \leftarrow$ top $K$ CFG rules used
for $r = 2$ to $R$ do
    $S \leftarrow S \cup \{$observed 1-rule extensions of $F \in \mathcal{F}_{\langle r-1,K \rangle}\}$
    $\mathcal{F}_{\langle r,K \rangle} \leftarrow$ top $K$ elements of $\mathcal{F}_{\langle r-1,K \rangle} \cup S$
end for
```

Pseudo-code is provided in Algorithm 1.[4]

This incremental approach is appealing for two reasons. Firstly, our approach tempers the growth of intermediate rankings $\mathcal{F}_{\langle r,K \rangle}$. Secondly, we have two tunable parameters R and K, which can be thought of as weakly being related to the base measure and concentration parameter of (Post and Gildea, 2009; Cohn et al., 2010). Note that by thresholding at every iteration, we enforce sparsity.

4 Experiments

We view grammaticality judgment as a binary classification task: is a sequence of words grammatical or not? We evaluate on two tasks of differing granularity: the first, a coarse-grain classification, follows Cherry and Quirk (2008); the other, a fine-grain analogue, is built upon Foster and Andersen (2009).

4.1 Datasets

For the coarse-grained task, we use the BLLIP[5]-inspired dataset, as in Post (2011), which discriminates between BLLIP sentences and Kneyser-Ney trigram generated sentences (of equal length). Grammatical and ungrammatical examples are given in 1 and 2 below, respectively:

(1) The most troublesome report may be the August merchandise trade deficit due out tomorrow .

(2) To and , would come Hughey Co. may be crash victims , three billion .

For the fine-grained task we use a version of the BNC that has been automatically modified to be ungrammatical, via insertions, deletions or substitutions of grammatically important words. As has been argued in previous work, these automatically generated errors, simulate more realistic errors (Foster and Andersen, 2009). Example 3 gives an original sentence, with an italicized substitution error:

(3) The league 's promoters hope retirees and tourists will join die-hard fans like Mr. de Castro and pack *then* stands to see the seniors .

Both sets contain train/dev/test splits with an equal number of positive and negative examples, and all instances have an available gold-standard parse[6].

4.2 Models and Features

Algorithm 1 extracts common constructions, in the form of count-extracted fragments. To test the efficacy of these fragments, we construct and experiment with various discriminative models.

Given count-extracted fragments obtained from EXTRACTFRAGMENTS(R, K), it is easy to define a feature vector: for each query, there is a binary feature indicating whether a particular extracted fragment occurs in its gold-standard parse. These count-extracted features, along with the sentence length, define the first model, called COUNT.

Although our extracted fragments may help identify grammatical constructions, capturing ungrammatical constructions may be difficult, since we do not parse with our fragments. Thus, we created two augmented models, COUNT+LEX and COUNT+CFG, which built upon and extended COUNT. COUNT+LEX included all preterminal and lexical items. For COUNT+CFG, we included a binary feature for every rule that was used in the most likely parse of a query sentence, according to a PCFG[7].

Following Post (2011), we train an ℓ-2 regularized SVM using `liblinear`[8] (Fan et al., 2008) per model. We optimized the models on dev data, letting the smoothing parameter be 10^m, for integral $m \in [-4, 2]$: 0.1 was optimal for all models.

[4] Code is available at: `cs.jhu.edu/~ferraro`.

[5] LDC2000T43

[6] We parsed all sentences with the Berkeley parser (Petrov et al., 2006).

[7] We used the Berkeley grammar/parser (Petrov et al., 2006) in `accurate` mode; all other options were their default values.

[8] `csie.ntu.edu.tw/~cjlin/liblinear/`

Task	COUNT	COUNT+LEX	COUNT+CFG
coarse	86.3	86.8	**88.3**
fine	62.9	64.3	**67.0**

(a) Our count-based models, with $R = 15$, $K = 50k$.

Task	3	5	10	15
coarse	**89.2**	89.1	88.6	88.3
fine	**67.9**	67.2	67.2	67.0

(b) Performance of COUNT+CFG, with $K = 50k$ and varying R.

Table 1: Development accuracy results.

Method	coarse	fine
COUNT+CFG, $R = 3$	**89.1**	**67.2**
COUNT+CFG, $R = 15$	88.2	66.6
bigram	68.4	61.4
CFG	86.3	64.5
TSG	**89.1**	67.0

Table 2: Classification accuracy on test portions for both **coarse** and **fine**, with $K = 50k$. Chance is 50% for each task.

Our three models all have the same two tunable parameters, R and K. While we initially experimented with $R = 31, K \in \{50k, 100k\}$ — in order to be comparable to the size of Post (2011)'s extracted TSGs — we noticed that very few, if any, fragments of size greater than 15 are able to survive thresholding. Dev experimentation revealed that $K = 50k$ and $100k$ yielded nearly the same results; for brevity, we report in Table 1a dev results for all three models, with $R = 15, K = 50k$. The differences across models was stark, with COUNT+CFG yielding a two point improvement over COUNT on **coarse**, but a four point improvement on **fine**. While COUNT+LEX does improve upon COUNT, on both tasks it falls short of COUNT+CFG. These differences are not completely surprising: one possible explanation is that the PCFG features in COUNT+CFG yield useful negatively-biased features, by providing *a* generative explanation. Due to the supremacy of COUNT+CFG, we solely report results on COUNT+CFG.

In Table 1b, we also examine the effect of extracted rule depth on dev classification accuracy, where we fix $K = 50k$ and vary $R \in \{3, 5, 10, 15\}$, where the best results are achieved with $R = 3$. We evaluate two versions of COUNT+CFG: one with $R = 3$ and the other with $R = 15$ ($K = 50k$ for both).

5 Results and Fragment Analysis

We build on Post (2011)'s results and compare against bigram, CFG and TSG baselines. Each baseline model is built from the same ℓ-2 regularized SVM as above, and each is optimized on dev data. For the bigram baseline, the binary features correspond with whether a particular bigram appears in an instance, while the CFG baseline is simply the augmentation feature set used for COUNT+CFG. For the TSG baseline, the binary features correspond with whether a particular fragment is used in the most probable derivation of each input sentence (using Post's Bayesian TSGs). All baselines use the sentence length as a feature as well.

The results on the test portions of each dataset are given in Table 2. When coupled with the best parse output, our counting method was able to perform on par with, and even surpass, Post's TSGs. The simpler model ($R = 3$) ties TSG performance on **coarse** and exceeds it by two-tenths on **fine**; the more complex model ($R = 15$) gets within a point on **coarse** and four-tenths on **fine**. Note that both versions of COUNT+CFG surpass the CFG baseline on both sets, indicating that (1) encoding deeper structure, even without an underlying probabilistic model, is useful for grammaticality classifications, and (2) this deeper structure can be achieved by a simple counting scheme.

As PCFG output comprises a portion of our feature set, it is not surprising that a number of the most discriminative positive and negative features, such as flat NP and VP rules not frequent enough to survive thresholding, were provided by the CFG parse. While this points out a limitation of our non-adaptive thresholding, note that even among the highest weighted features, PCFG and count-extracted features were interspersed. Further, considering that both versions of COUNT+CFG outperformed CFGs, it seems our method adds discriminative power to the CFG rules.

	(a) Coarse			(b) Fine	
	Grammatical	**Ungrammatical**		**Grammatical**	**Ungrammatical**
1	(S NP VP (. .))	(S NP (VP (VBP are) PP))	10	(SBAR (IN if) S)	(SBAR (S VP))
2	(S (S (VP VBG NP)) VP)	(VP VBZ (S VP))	11	(NP (DT these) NNS)	(SBAR DT (S NP VP))
3	(SBAR (IN while) S)	(SBAR (S VP))	12	(VP (VBG being) VP)	(S (VP VB NP))
4	(VP (VBD called) S)	(VP VBN (S VP))	13	(PP IN (S NP (VP VBG NP)))	(S (VP VBZ NP))
5	(VP (VB give) NP NP)	(NP (NP JJ NN) SBAR)	14	(S (VP VBG VP))	(VP VB (S VP))
6	(NP NNP NNP NNP (NNP Inc.))	(VP NN (PP IN NP))	15	(PP IN (SBAR (IN whether) S))	(S (VP VBP VP))
7	(PP (IN with) (S NP VP))	(S (VP MD VP))	16	(VP (VBD had) (VP VBN S))	(S NP (VP (VBD said)))
8	(SBAR (IN for) (S NP (VP (TO to) VP)))	(SBAR (S (NP NNS) VP))	17	(VP MD (VP VB NP (PP IN NP) PP))*	(PP (PP IN NP) (CC and) PP)*
9	(PRN (-LRB- -LRB-) NP (-RRB- -RRB-))*	(S (ADJP JJ))*	18	(NP (DT no) NNS)*	(PP (IN As) NP)*

Table 3: Most discriminative count-based features for COUNT+CFG on both tasks. For comparability to Post (2011), $R = 15, K = 50k$, are shown. Asterisks (*) denote fragments hand-selected from the top 30.

Table 5 presents top weighted fragments from COUNT+CFG on both **coarse** and **fine**, respectively. Examining useful grammatical features across tasks, we see a variety of fragments: though our fragments heavily weight simple structure such as proper punctuation (ex. 1) and parentheticals (ex. 9), they also capture more complex phenomena such as lexical argument descriptions (e.g., *give*, ex. 5). Our extracted fragments also describe common constructions and transitions (e.g., 3, 8 and 15) and involved verb phrases (e.g., gerunds in 2 and 14, perfectives in 16, and modals in 17).

Though for both tasks some ungrammatical fragments easily indicate errors, such as sentence fragments (e.g., example 6) or repeated words (ex. 11), in general the analysis is more difficult. In part, this is because, when isolated from errors, one may construct grammatical sentences that use some of the highest-weighted ungrammatical fragments. However, certain errors may force particular rules to be inappropriately applied when acquiring the gold-standard parse. For instance, example 10 typically coordinates with larger VPs, via auxiliary verbs or expletives (e.g., *it*). Affecting those crucial words can significantly change the overall parse structure: consider that in “said *it* is too early…,” *it* provides a crucial sentential link; without it, “is too early” may be parsed as a sentence, and then glued on to the former part.

6 Conclusion

In this work, we further examined TSGs as useful judges of grammaticality for written English. Using an iterative, count-based approach, along with the most likely PCFG parse, we were able to train a discriminative classifier model — COUNT+CFG — that surpassed the PCFG’s ability to judge grammaticality, and performed on par with Bayesian-TSGs. Examining the highest weighted features, we saw that complex structures and patterns encoded by the count-based TSGs proved discriminatively useful. This suggests new, simpler avenues for fragment learning, especially for grammaticality judgments and other downstream tasks.

Acknowledgements Thank you to the reviewers for helpful feedback, and thanks to Johns Hopkins HLTCOE for providing support. Any opinions expressed in this work are those of the authors.

References

R. Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, pages 37–44. Association for Computational Linguistics.

Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent svms. *Proceeding of Association for Machine Translation in the America (AMTA-2008)*.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in bayesian tree substitution grammars. In *Proceedings of ACL (short papers)*, pages 225–230, Stroudsburg, PA, USA. Association for Computational Linguistics.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Stroudsburg, PA, USA. Association for Computational Linguistics.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096, December.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, June.

Jennifer Foster and Oistein E. Andersen. 2009. GenERRate: generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–90.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21 – 54.

Joshua Goodman. 1996. Efficient algorithms for parsing the dop model. In *Proceedings of EMNLP*, pages 143–152.

A.K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? *Natural language parsing*, pages 206–250.

Percy Liang, Michael .I. Jordan, and Dan Klein. 2010. Type-based MCMC. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, et al. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of NAACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL-ICCL*, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of ACL-IJCNLP (short papers)*, pages 45–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Post. 2011. Judging grammaticality with tree substitution grammar derivations. In *Proceedings of ACL (short papers)*, pages 217–222, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Wagner, J. Foster, and J. van Genabith. 2009. Judging grammaticality: Experiments in sentence classification. *CALICO Journal*, 26(3):474–490.

Sze-Meng Jojo Wong and Mark Dras. 2010. Parser features for sentence grammaticality classification. In *Proceedings of the Australasian Language Technology Association Workshop*.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for Data-Oriented Parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.

Willem Zuidema. 2007. Parsimonious data-oriented parsing. In *Proceedings of EMNLP-CoNLL*, pages 551–560.