Explicit and Implicit Syntactic Features for Text Classification

Matt Post[1] and **Shane Bergsma**[1,2]
[1]Human Language Technology Center of Excellence
[2]Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD

Abstract

Syntactic features are useful for many text classification tasks. Among these, tree kernels (Collins and Duffy, 2001) have been perhaps the most robust and effective syntactic tool, appealing for their empirical success, but also because they do not require an answer to the difficult question of *which* tree features to use for a given task. We compare tree kernels to different explicit sets of tree features on five diverse tasks, and find that explicit features often perform as well as tree kernels on accuracy and always in orders of magnitude less time, and with smaller models. Since explicit features are easy to generate and use (with publicly available tools), we suggest they should always be included as baseline comparisons in tree kernel method evaluations.

1 Introduction

Features computed over parse trees are useful for a range of discriminative tasks, including authorship attribution (Baayen et al., 1996), parse reranking (Collins and Duffy, 2002), language modeling (Cherry and Quirk, 2008), and native-language detection (Wong and Dras, 2011). A major distinction among these uses of syntax is how the features are represented. The **implicit approach** uses tree kernels (Collins and Duffy, 2001), which make predictions with inner products between tree pairs. These products can be computed efficiently with a dynamic program that produces weighted counts of all the shared tree fragments between a pair of trees, essentially incorporating all fragments without representing any of them explicitly. Tree kernel approaches have been applied successfully in many areas of NLP (Collins and Duffy, 2002; Moschitti, 2004; Pighin and Moschitti, 2009).

Tree kernels were inspired in part by ideas from Data-Oriented Parsing (Scha, 1990; Bod, 1993), which was in turn motivated by uncertainty about which fragments to include in a grammar. However, manual and automatic approaches to inducing tree fragments have recently been found to be useful in an **explicit approach** to text classification, which employs specific tree fragments as features in standard classifiers (Post, 2011; Wong and Dras, 2011; Swanson and Charniak, 2012). These feature sets necessarily represent only a small subset of all possible tree patterns, leaving open the question of what further gains might be had from the unusued fragments.

Somewhat surprisingly, explicit and implicit syntactic features have been explored largely independently. Here, we compare them on a range of classification tasks: (1,2) grammatical classification (is a sentence written by a human?), (3) question classification (what type of answer is sought by this question?), and (4,5) native language prediction (what is the native language of a text's author?).

Our main contribution is to show that an explicit syntactic feature set performs as well or better than tree kernels on each tested task, and in orders of magnitude less time. Since explicit features are simple to generate (with publicly available tools) and flexible to use, we recommend they be included as baseline comparisons in tree kernel method evaluations.

2 Experimental setup

We used the following feature sets:

N-grams All unigrams and bigrams.[1]

[1]Experiments with trigrams did not show any im-

CFG rules Counts of depth-one context-free grammar (CFG) productions obtained from the Berkeley parser (Petrov et al., 2006).

C&J features The parse-tree reranking feature set of Charniak and Johnson (2005), extracted from the Berkeley parse trees.

TSG features We also parsed with a Bayesian tree substitution grammar (Post and Gildea, 2009, TSG)[2] and extracted fragment counts from Viterbi derivations.

We build classifiers with LIBLINEAR[3] (Fan et al., 2008). We divided each dataset into training, dev, and test sets. We then trained an L2-regularized L1-loss support vector machine (`-s 3`) with a bias parameter of 1 (`-B 1`), optimizing the regularization parameter (`-c`) on the dev set over the range $\{0.0001 \ldots 100\}$ by multiples of 10. The best model was then used to classify the test set. A sentence length feature was included for every sentence.

For tree kernels, we used SVM-light-TK[4] (Moschitti, 2004; Moschitti, 2006) with the default settings (`-t 5 -D 1 -L 0.4`),[5] which also solves an L2-regularized L1-loss SVM optimization problem. We tuned the regularization parameter (`-c`) on the dev set in the same manner as described above, providing 4 GB of memory to the kernel cache (`-m 4000`).[6] We used *subset tree kernels*, which compute the similarity between two trees by implicitly enumerating all possible fragments of the trees (in contrast with *subtree kernels*, where all fragments fully extend to the leaves).

3 Tasks

Table 1 summarizes our datasets.

3.1 Coarse grammatical classification

Our first comparison is coarse grammatical classification, where the goal is to distinguish between human-written sentences and "pseudo-negative" sentences sampled from a trigram language model constructed from in-domain data (Okanohara and Tsujii, 2007). Cherry and Quirk (2008) first applied syntax to this task, learning weighted parameters for a CFG with a latent SVM. Post (2011) found further improvements with fragment-based representations (TSGs and C&J) with a regular SVM. Here, we compare their results to kernel methods. We repeat Post's experiments on the BLLIP dataset,[7] using his exact data splits (Table 2). To our knowledge, tree kernels have not been applied to this task.

	train	dev	test
Coarse grammaticality (BLLIP)			
sentences	100,000	6,000	6,000
Fine grammaticality (PTB)			
sentences	79,664	3,978	3,840
Question classification (TREC-10)			
sentences	4,907	545	500
Native language (ICLE; 7 languages)			
documents	490	105	175
sentences	17,715	3,968	6,777
Native language (ACL; 5 languages)			
documents	987	195	185
sentences	146,257	28,139	28,403

Table 1: Datasets.

system	accuracy	CPU time
Chance	50.0	-
N-gram	68.4	minutes
CFG	86.3	minutes
TSG	89.8	minutes
C&J	**92.9**	an hour
SVM-TK	91.0	a week

Table 2: Coarse grammaticality. CPU time is for classifier setup, training, and testing.

3.2 Fine grammatical classification

Real-world grammaticality judgments require much finer-grained distinctions than the coarse ones of the previous section (for example, marking dropped determiners or wrong verb inflections). For this task, we too positive examples from all sentences of sections 2–21 of the WSJ portion of the Penn Treebank (Marcus et al., 1993). Negative examples were created by inserting one or two errors

provement.

[2]`github.com/mjpost/dptsg`

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

[4]`disi.unitn.it/moschitti/Tree-Kernel.htm`

[5]Optimizing SVM-TK's decay parameter (`-L`) did not improve test-set accuracy, but did increase training time (squaring the number of hyperparameter combinations to evaluate), so we stuck with the default.

[6]Increased from the default of 40 MB, which halves the running time.

[7]LDC Catalog No. LDC2000T43

system	accuracy	CPU time
Wong & Dras	60.6	-
Chance	50.0	-
N-gram	61.4	minutes
CFG	64.5	minutes
TSG	67.0	minutes
C&J	**71.9**	an hour
SVM-TK	67.8	weeks

Table 3: Fine-grained classification accuracy (the Wong and Dras (2010) score is the highest score from the last column of their Table 3).

system	accuracy	CPU time
Pighin & Moschitti	86.6	-
Bigram	73.2	seconds
CFG	**90.0**	seconds
TSG	85.6	seconds
C&J	89.6	minutes
SVM-TK	87.7	twenty min.

Table 4: Question classification (6 classes).

into the parse trees from the positive data using GenERRate (Foster and Andersen, 2009). An example sentence pair is *But the ballplayers disagree[ing]*, where the negative example incorrectly inflects the verb. Wong and Dras (2010) reported good results with parsers trained separately on the positive and negative sides of the training data and classifiers built from comparisons between the CFG productions of those parsers. We obtained their data splits (described as *NoisyWSJ* in their paper) and repeat their experiments here (Table 3).

3.3 Question Classification

We look next at question classification (QC). Li and Roth (2002) introduced the TREC-10 dataset,[8] a set of questions paired with labels that categorize the question by the type of answer it seeks. The labels are organized hierarchically into six (coarse) top-level labels and fifty (fine) refinements. An example question from the ENTY/animal category is *What was the first domesticated bird?*. Table 4 contains results predicting just the coarse labels. We compare to Pighin and Moschitti (2009), and also repeat their experiments, finding a slightly better result for them.

[8]`cogcomp.cs.illinois.edu/Data/QA/QC/`

system	sent.	voting	whole
Wong & Dras	-	-	80.0
Style	42.0	75.3	**86.8**
CFG	39.5	73.2	83.7
TSG	38.7	72.1	83.2
C&J	**42.9**	76.3	86.3
SVM-TK	40.7	69.5	-
Style	42.5	65.3	83.7
CFG	39.2	52.6	**86.3**
TSG	40.4	56.8	84.7
C&J	**49.2**	66.3	81.1
SVM-TK	42.1	52.6	-

Table 5: Accuracy on ICLE (7 languages, top) and ACL (five, bottom) datasets at the sentence and document levels. All documents were signature-stylized (§3.4).

We also experimented with the refined version of the task, where we directly predict one of the fifty refined categories, and found nearly identical relative results, with the best explicit feature set (CFG) returning an accuracy of 83.6% (in seconds), and the tree kernel system 69.8% (in an hour). For reference, Zhang and Lee (2003) report 80.2% accuracy when training on the full training set (5,500 examples) with an SVM and bag-of-words features.[9]

3.4 Native language identification

Native language identification (NLI) is the task of determining a text's author's native language. This is usually cast as a document-level task, since there are often not enough cues to identify native languages at smaller granularities. As such, this task presents a challenge to tree kernels, which are defined at the level of a single parse tree and have no obvious document-level extension. Table 5 therefore presents three evaluations: (a) sentence-level accuracy, and document-level accuracy from (b) sentence-level voting and (c) direct, whole-document classification.

We perform these experiments on two datasets. In order to mitigate topic bias[10] and other problems that have been reported with

[9]Pighin and Moschitti (2009) did not report results on this version of the task.

[10]E.g., when we train with all words, the keyword 'Japanese' is a strong indicator for Japanese authors, while 'Arabic' is a strong indicator for English ones.

the ICLE dataset (Tetreault et al., 2012),[11] we preprocessed each dataset into two signature-stylized versions by replacing all words not in a stopword list.[12] The first version replaces non-stopwords with word classes computed from surface-form signatures,[13] and the second with POS tags.[14] N-gram features are then taken from both stylized versions of the corpus.

Restricting the feature representation to be topic-independent is standard-practice in stylometric tasks like authorship attribution, gender identification, and native-language identification (Mosteller and Wallace, 1984; Koppel et al., 2003; Tomokiyo and Jones, 2001).

3.4.1 ICLE v.2

The first dataset is a seven-language subset of the *International Corpus of Learner English, Version 2* (ICLE) (Granger et al., 2009), which contains 3.7 million words of English documents written by people with sixteen different native languages. Table 1 contains scores, including one reported by Wong and Dras (2011), who used the CFG and C&J features, and whose data splits we mirror.[15]

3.4.2 ACL Anthology Network

We also experimented with native language classification on scientific documents using a version of the ACL Anthology Network (Radev et al., 2009, AAN) annotated for experiments in stylemetric tasks, including a native/non-native author judgment (Bergsma et al., 2012). For NLI, we further annotated this dataset in a semi-automatic fashion for the five most-common native languages of ACL authors in our training era: English, Japanese, German, Chinese, and French. The annotation heuristics, designed to favor precision over recall, provided annotations for 1,959 of 8,483 papers (23%) in the 2001–2009 AAN.[16]

[11]Including prompts, characters, and special tokens that correlate strongly with particular outcomes.

[12]The stopword list contains the set of 524 SMART-system stopwords used by Tomokiyo and Jones (2001), plus punctuation and Latin abbreviations.

[13]For example, suffix and capitalization.

[14]Via CRFTagger (Phan, 2006).

[15]Tetreault et al. reported accuracies up to 90.1 in a cross-validation setting that isn't directly comparable.

[16]Details and data at `old-site.clsp.jhu.edu/~sbergsma/Stylo/`.

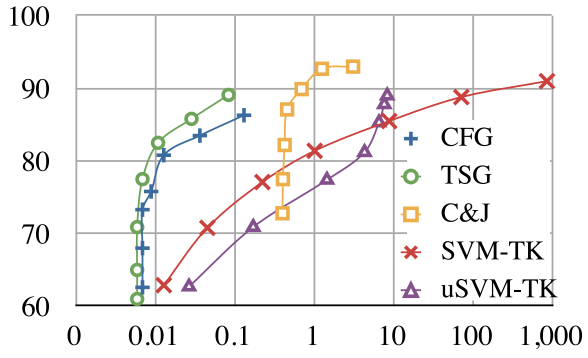

Figure 1: Training time (1000s of seconds) vs. test accuracy for coarse grammaticality, plotting test scores from models trained on 100, 300, 1k, 3k, 10k, 30k, and 100k instances.

4 Discussion

Syntactic features improve upon the n-gram baseline for all tasks except whole-document classification for ICLE. Tree kernels are often among the best, but always trail (by orders of magnitude) when runtime is considered. Constructing the multi-class SVM-TK models for the NLI tasks in particular was computationally burdensome, requiring cpu-months of time. The C&J features are similarly often the best, but incur a runtime cost due to the large models. CFG and TSG features balance performance, model size, and runtime. We now compare these approaches in more depth.

4.1 Training time versus accuracy

Tree kernel training is quadratic in the size of the training data, and its empirical slowness is known. It is informative to examine learning curves to see how the time-accuracy tradeoffs extrapolate. We compared models trained on the first 100, 300, 1k, 3k, 10k, 30k, and 100k data points of the coarse grammaticality dataset, split evenly between positive and negative examples (Figure 1). SVM-TK improves over the TSG and CFG models in the limit, but at an extraordinary cost in training time: 100k training examples is already pushing the bounds of practicality for tree kernel learning, and generating curve's next point would require several *months* of time. Kernel methods also produce large models that result in slow *test*-time performance, a problem dubbed the "curse of kernelization" (Wang et al., 2010).

Approximate kernel methods designed to scale to large datasets address this (Severyn and Moschitti, 2010). We investigated the uSVM-TK toolkit,[17] which enables tuning the tradeoff between training time and accuracy. While faster than SVM-TK, its performance was never better than explicit methods along both dimensions (time and accuracy).

4.2 Overfitting

Overfitting is also a problem for kernel methods. The best models often had a huge number of support vectors, achieving near-perfect accuracy on the training set but making many errors on the dev. and test sets. On the ICLE task, close to 75% of all the training examples were used as support vectors. We found only half as many support vectors used for the explicit representations, implying less error (Vapnik, 1998), and saw much lower variance between training and testing performance.

4.3 Which fragments?

Our findings support the observations of Cumby and Roth (2003), who point out that kernels introduce a large number of irrelevant features that may be especially harmful in small-data settings, and that, when possible, it is often better to have a set of explicit, relevant features. In other words, it is better to have the *right* features than *all* of them. Tree kernels provide a robust, efficiently-computable measure of comparison, but they also skirt the difficult question, *Which fragments?*

So what are the "right" features? Table 6) presents an intuitive list from the coarse grammaticality task: phenomena such as balanced parenthetical phrases and quotations are associated with grammaticality, while small, flat, abstract rules indicate samples from the n-gram model. Similar intuitive results hold for the other tasks. The immediate interpretability of the explicit formalisms is another advantage, although recent work has shown that weights on the implicit features can also be obtained after a kind of linearization of the tree kernel (Pighin and Moschitti, 2009).

Ultimately, which features matter is task-dependent, and skirting the question is advantageous in many settings. But it is also encouraging that methods for selecting fragments and other tree features work so well, yielding quick, light-weight models that contrast with the heavy machinery of tree kernels.

[17] disi.unitn.it/~severyn/code.html

(TOP (S " S , " NP (VP (VBZ says) ADVP) .))
(FRAG (X SYM) VP .)
(PRN (-LRB- -LRB-) S (-RRB- -RRB-))
(PRN (-LRB- -LRB-) NP (-RRB- -RRB-))
(S NP VP .)
(NP (NP DT CD (NN %)) PP)
(NP DT)
(PP (IN of))
(TOP (NP NP PP PP .))
(NP DT JJ NNS)

Table 6: The highest- and lowest-weighted TSG features (coarse grammaticality).

5 Conclusion

Tree kernels provide a robust measure of comparison between trees, effectively making use of all fragments. We have shown that for some tasks, it is sufficient (and advantageous) to instead use an explicitly-represented subset of them. In addition to their flexibility and interpetability, explicit syntactic features often outperformed tree kernels in accuracy, and even where they did not, the cost was *multiple orders of magnitude increase* in both training and testing time. These results were consistent across a range of task types, dataset sizes, and classification arities (binary and multiclass).

There are a number of important caveats. We explored a range of data settings, but there are many others where tree kernels have been proven useful, such as parse tree reranking (Collins and Duffy, 2002; Shen and Joshi, 2003), sentence subjectivity (Suzuki et al., 2004), pronoun resolution (Yang et al., 2006), relation extraction (Culotta and Sorensen, 2004), machine translation evaluation (Liu and Gildea, 2005), predicate-argument recognition, and semantic role labeling (Pighin and Moschitti, 2009). There are also tree kernel variations such as dependency tree kernels (Culotta and Sorensen, 2004) and shallow semantic tree kernels (Moschitti et al., 2007). These variables provide a rich environment for future work; in the meantime, we take these results as compelling motivation for the continued development of explicit syntactic features (both manual and automatically induced), and suggest that such features should be part of the baseline systems on applicable discriminative NLP tasks.

References

Harald Baayen, Hans Van Halteren, and Fiona Tweedie. 1996. Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121.

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proc. of NAACL-HLT*, pages 327–337, Montréal, Canada, June. Association for Computational Linguistics.

Rens Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proc. of ACL*, Columbus, Ohio, USA, June.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*, pages 173–180, Ann Arbor, Michigan, USA, June.

Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent SVMs. In *Proc. of AMTA*, Waikiki, Hawaii, USA, October.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proc. of NIPS*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proc. of ACL*, pages 173–180, Philadelphia, Pennsylvania, USA, July.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*, pages 423–429.

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.

Jennifer Foster and Øistein 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.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. The International Corpus of Learner English. Version 2. Handbook and CD-Rom.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2003. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proc. of COLING*, pages 1–7.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):330.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proc. of ACL*, pages 776–783, Prague, Czech Republic, June.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proc. of ACL*.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proc. of EACL*, volume 6, pages 113–120.

Frederick Mosteller and David L. Wallace. 1984. *Applied Bayesian and Classical Inference: The Case of the Federalist Papers*. Springer-Verlag.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proc. of ACL*, Prague, Czech Republic, June.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*, Sydney, Australia, July.

Xuan-Hieu Phan. 2006. CRFTagger: CRF English POS Tagger. `crftagger.sourceforge.net`.

Daniele Pighin and Alessandro Moschitti. 2009. Reverse engineering of tree kernel feature spaces. In *Proc. of EMNLP*, pages 111–120, Singapore, August.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proc. of ACL (short paper track)*, Suntec, Singapore, August.

Matt Post. 2011. Judging grammaticality with tree substitution grammar derivations. In *Proc. of ACL*, Portland, Oregon, USA, June.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *Proc. of ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, pages 54–61.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de neerlandistiek*, pages 7–22, Almere, the Netherlands. De Vereniging.

Aliaksei Severyn and Alessandro Moschitti. 2010. Large-scale support vector learning with structural kernels. In *Proc. of ECML/PKDD*, pages 229–244.

Libin Shen and Aravind K. Joshi. 2003. An SVM-based voting algorithm with application to parse reranking. In *Proc. of CoNLL*, pages 9–16.

Jun Suzuki, Hideki Isozaki, and Eisaku Maeda. 2004. Convolution kernels with feature selection for natural language processing tasks. In *Proc. of ACL*, pages 119–126.

Benjamin Swanson and Eugene Charniak. 2012. Native language detection with tree substitution grammars. In *Proc. of ACL (short papers)*, pages 193–197, Jeju Island, Korea, July.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proc. of COLING*, pages 2585–2602, Mumbai, India, December.

Laura Mayfield Tomokiyo and Rosie Jones. 2001. You're not from 'round here, are you? Naive Bayes detection of non-native utterances. In *Proc. of NAACL*.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Zhuang Wang, Koby Crammer, and Slobodan Vucetic. 2010. Multi-class pegasos on a budget. In *ICML*, pages 1143–1150.

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

Sze-Meng Jojo Wong and Mark Dras. 2011. Exploiting parse structures for native language identification. In *Proc. of EMNLP*, pages 1600–1610, Edinburgh, Scotland, UK., July.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. of Coling-ACL*, pages 41–48.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 26–32, New York, NY, USA. ACM.