Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers

Mark Hepple
Department of Computer Science, University of Sheffield, Regent Court,
211 Portobello Street, Sheffield S1 4DP, UK [hepple@dcs.shef.ac.uk]

Abstract

This paper addresses the rule-based POS tagging method of Brill, and questions the importance of rule interactions to its performance. Adopting two assumptions that serve to exclude rule interactions during tagging and training, we arrive at some variants of Brill's approach that are instances of *decision list* models. These models allow for both rapid training on large data sets and rapid tagger execution, giving tagging accuracy that is comparable to, or better than the Brill method.

1 Introduction

Part-of-speech (POS) tagging is the task of assigning to each word in a sentence a *tag* indicating its lexical syntactic category, such as noun or verb. POS tagging of text is required for subsequent processes in many systems, e.g. syntactic parsing. A number of alternative models and methods for tagging have been explored, most particularly with a view to improving tagging accuracy, including: hidden Markov models (Church, 1988; Charniak et al., 1993; Cutting et al., 1992), rule-based methods (Brill, 1995), maximum entropy methods (Ratnaparkhi, 1996), memory-based methods (Daelemans et al., 1996), amongst others.

This paper addresses the rule-based POS tagging approach of Brill (1993; 1995), which learns language models that consist of a sequence of transformation rules (TRs) which capture contextual factors in predicting correct assignments. The approach allows certain interactions between rules uses, whereby a change effected by one may affect whether or not another may subsequently fire. Such interactions have been assumed important for the approach in regard to its success on the POS tagging task.

In this paper, we explore the possibility that such interactions are *not* empirically important for Brill's approach, at least as applied to POS tagging. To this end, we introduce two assumptions, called *independence* and *commitment*, which serve to exclude rule interactions, and develop two variants of Brill's approach that realise these assumptions. These methods turn out to be instances of *decision list* models, a standard approach within the machine learning field (Rivest, 1987). These models allow for training and execution algorithms that give much improved performance in terms of speed, with an impact on tagging accuracy that ranges from slight degradation to small improvement. These results serve to shed light on the importance of rule interaction to the performance of Brill's original model.

2 Brill's Tagging Approach

Brill's supervised learning model "transformation-based error-driven learning" works as follows. The training data is correctly annotated text. The corresponding raw text is fed through an initial-state annotator, which makes a more or less well-informed initial guess of how the text

[0]I am indebted to Rob Gaizauskas for discussions which helped greatly in the development of the ideas in this paper.

should be annotated. This initial annotation is compared to the training data as a basis for learning a sequence of TRs, which are context dependent 'correction' rules, that apply in sequence to modify the initial annotation to better approximate the training data. The trained system consists of the initial-state annotator together with the ordered sequence of TRs, which can be applied to unseen text.

For POS tagging, the initial-state annotator assigns each known word its most-probable tag from amongst those listed in a lexicon, as determined from some training corpus. The initial tagging of unknown words can be handled in a number of ways, using clues such as affixes and capitalisation. However, we shall not dwell on the treatment of unknown words in this paper. The initial annotation is compared with the training data to identify TRs that improve accuracy by making specific context-bound corrections steps, e.g. replacing tag X with Y, provided tag Z appears in some nearby position (where Y is also a lexical tag of the word).

The space of possible TRs is fixed by stating a set of rule 'templates', which are essentially underspecified TRs. For example, one template provides the pattern 'change tag A to B where the previous word has tag C' (where A,B,C are unspecified). Another allows the change if tag C is assigned to either of the two preceding words, etc. In the TRs that are learned, these unspecified values (A,B,C) are instantiated to specific parts of speech. Rules can also require specific words, rather than just tags, to appear in context.

TRs are learned as an ordered sequence. At each stage, the next rule adopted is the one that gives the best *net* improvement in tagging accuracy (since rules can both effect corrections *and* introduce errors). This rule is then applied to the current tag state of the training set (which is initially the initial-state assignment), and the next rule sought. This process terminates when the improvement falls to some prespecified threshold.

An interesting feature of this approach is that it allows certain interactions between rules, with possible beneficial effects. A change effected by one rule may cause the context requirement of another rule to be satisfied allowing it to fire. For example, given rules "change b to c if previous tag is a" and "change d to e if previous tag is c" and a sequence *abd*, the first rule may fire to give *acd*, allowing the second rule to fire giving *ace*. A second form of interaction is where the tag substituted by one rule is itself subject to change by another rule. For example, given rules "change b to c if previous tag is a" and "change c to d if next tag is e" and a sequence *abe*, the first rule can fire giving *ace*, allowing the second rule to fire, to again affect the second position, giving *ade*. Such rule interactions have been seen as an important feature, allowing what Ramshaw and Marcus (1994) call 'leveraging', as in "leveraging of partial solutions between neighboring instances", i.e. so that corrections effected by earlier rules in sequence allow more accurate operation of later rules (both as part of learning, and in execution).

An advantage of Brill's approach is that its learned model is quite compact, consisting of a few hundred rules, that can be directly inspected (c.f. the thousands of contextual probabilities of a HMM tagger). As discussed in Ramshaw and Marcus (1994), the approach is resistant to overtraining effects. In Brill's experiments (Brill, 1995), training on 600K tokens of the PTB tagged *Wall Street Journal* corpus under the 'closed vocabulary assumption' (where there are no unknown words) gave tagging accuracy of 97.2%. Without this assumption, where performance also depends on the handling unknown words, the score was 96.6%. These results were state-of-the-art at the time, but have since been surpassed by some statistical approaches and 'voting' systems that combine multiple taggers (Ratnaparkhi, 1996; van Halteren et al., 1998).

Problems of Brill's approach include, firstly, the speed of tagging, which was found to be significantly slower than HMM-based competitors. However, Roche and Schabes (1995) show how to compile a rule-based tagger to a finite-state transducer, giving very fast execution. Secondly, there is the cost in time of training, which for larger-sized training sets (e.g. 600K tokens) will take something around a day or more to complete (using Brill's own implementation). Ramshaw and Marcus (1994) propose an faster 'incremental' training algorithm (which avoids rescanning the corpus for each rule that is learned by using lists of pointers to link rules to the sites where they apply), but note that its memory requirements are so high as to limit its applicability. Samuel (1998) proposes a 'lazy' version of transformation-based learning, a Monte Carlo variant of the standard method, and applies it to dialogue act tagging. The μ-TBL system of Lager (1999) is an efficient Prolog implementation of transformation-based learning, which trains in shorter time than Brill's own implementation (by an order of magnitude, for the tasks reported). μ-TBL also implements 'lazy' learning, and Lager's results indicate comparable tagging accuracy for the lazy and standard methods as applied to POS tagging.

3 Is Rule Interaction Important?

The starting point for the work in this paper is questioning the assumption that rule interaction is important to the performance of Brill's method on POS tagging. Consider the fact that, for PTB *Wall Street Journal* text, baseline performance under the closed vocabulary assumption, from assigning the most-probable tag to each word, is around 94.5%. A final tagging accuracy of around 97% indicates an improvement of around 1 in 40 tags being corrected. Given this 'spareness' of corrections, we might suspect that the frequency of one correction being appropriately close to enable another will be quite low.

The only hard results of which we are aware that bear on this issue are given by Ramshaw and Marcus (1994), who report that rule learning on a 50K token sample of the Brown Corpus involved changes at 3395 sites, of which for only 395 sites did the change depend on the action of more than one rule. These 395 sites account for about 0.8% of the training set, but note that this does translate to 0.8% of the resulting tagging accuracy since applying a rule can have either positive, negative or neutral consequences.

In the remainder of the paper, we will develop some rule-based tagging methods for which rule interaction is explicitly excluded. The performance of these models provides empirical evidence concerning the real importance of rule interaction to transformation-based POS tagging.

4 Independence and Commitment

Our view of rule interaction is embodied by two assumptions, we call *independence* and *commitment*, which together serve to exclude rule interactions. The commitment assumption regulates how rules are used, requiring that where a rule fires to change a tag, that tag may not subsequently be changed again (so we are 'committed' to the change). Independence is the assumption that the frequency with which an earlier rule will modify the context relevant to the firing of a later rule is sufficiently low that it can be ignored.

Individually, these assumptions still allow a considerable degree of freedom as to how a method might be specified. During tagging, for example, commitment leaves it open as to whether a rule sequence should be used by applying each rule in turn to the entire initial tag state or if they should be applied as a group to tag each token in turn. Either option allows the possibility that earlier tag replacement steps might affect later rule firings. Independence, however, instructs us to ignore such interactions, in which case the two options should give essentially equivalent results. We will pursue the second option, with rules being applied as a group to each token, which involves testing each rule in turn until the list is exhausted or one rule fires, in which case the remaining rules are ignored.

This 'one-shot' style of using rules is familiar in machine learning as an instance of a *decision list* model (Rivest, 1987). More specifically, given the character of the rules involved, we have a *propositional* decision list system.[1] A standard use of decision lists is

[1]The more expressive formalism of *first-order* de-

for classification tasks, i.e. assigning categories to examples, based on their (static) characteristics. The 'dynamic' character tagging makes it somewhat different from classification, a fact which might present difficulties for decision list learning were it not for the independence assumption.

In the Brill learning approach, rules are acquired in the same order as they are applied during tagging. Minimally adapting such an approach to a decision list use of TRs gives an instance of a *sequential covering algorithm*, i.e. the first rule learned 'covers' some part of the training data (that now 'belongs' to that rule), and then the next rule learned covers some part of what remains, and so on. Following Webb and Brkic (1993), we refer to this way of ordering rules as *appending*, i.e. new rules are appended to the end of the current rule list. Commonly in sequential rule learning, rules learned earlier are highly general, applying to many instances, whereas rules learned later are more specific, applying to fewer instances. The general rules learned earlier have a large *net* positive effect, but may get specific subsets of cases wrong. The Brill approach allows later rules to be learned to correct errors made by earlier rules, via sequential rule application. For a decision list treatment of tagging, this possibility does not arise.

Webb and Brkic (1993) advocates a variant of decision list learning in which rules acquired are *prepended*, i.e. added to the front of the current list, so that rules are applied in the reverse of the order in which they are learned. Such an approach looks promising for a decision list treatment of tagging as it will allow for mistakes made by earlier learned rules to be 'overridden' by later more specific rules that are prepended. In this regard, it may be useful to allow 'identity' TRs, which fire leaving the default tag unchanged, but serve to prevent the same position being modified by any subsequent rule in the list. We will consider training methods for rule-based tagging without rule interaction in both appending and prepending variants, which we will refer to as ICA ('independence and commitment, with appending') and ICP ('... with prepending'), respectively.

cision lists is used within the field Inductive Logic Programming, which in recent work has been applied to a range of NLP tasks. See, for example, (Cussens, 1997) for an ILP treatment of tagging, which is very different to the present approach.

5 The Training Algorithm

The independence and commitment assumptions have a consequence that is crucial in allowing a rapid and memory efficient training algorithm. This is that we can divide training into separate phases, each of which separately addresses only the learning of rules that modify a given single POS. Thus, in a phase learning rules that modify tag t, independence instructs us to ignore the fact that rules learned in some earlier (or later) phase might modify the contexts around positions tagged t. Also, the possibility that rules of other phases might modify some other tag to become t can be ignored, since commitment prevents this tag from being further modified by rules learned in the current phase.

Each phase of training can restrict its attention to only a limited portion of the training corpus, e.g. learning rules to modify tag t, we need only address relevant 'data points', which are those positions having initial tag t and at least one alternative lexical tag. The TRs that can apply at these positions are a fraction of those that could apply anywhere in the corpus. The approach allows for an 'incremental' training algorithm which does not suffer the space problems of Ramshaw and Marcus (1994). Initially, we must score for all transformations that apply at the data points of the phase. On identifying and adopting the best rule T, we do not need to recompute all rule scores, but can instead scan the data points to find those at which T fires, and update our existing record of rule scores in regard to the possible TRs that can fire at these locations. We can then immediately determine the next rule to adopt, and so on.

An informal sketch of the training algorithm is given i n This makes no provision for unknown words, since we make the 'closed vocabulary assumption' in our exper-

Load training corpus into memory (array), i.e. words + correct tag, and make initial tag assignment (assign most-probable tag to each word).
Then, for each tag x, learn list of rules that apply for this tag as follows:

1. Scan corpus, and record 'data points' for this phase, i.e. positions where default tag is x and word has at least one alternative tag (store these array offsets)
2. Compute initial TR correction scores. Firstly, scan for data points where tag x is incorrect, and at each score $+1$ for all possible TRs changing x to the correct tag (scores stored in a hash). Secondly, at each point where tag x is correct, score -1 for all possible TRs changing x to any lexical alternative — but *only* for rules already present in scores hash (i.e. rules effecting at least one correction).
3. Loop acquiring rules as follows:
   Scan scores hash for best rule T (exit loop if score not above threshold) and add to rule list. Update scores hash as follows: scan for data points where rule T fires, and update scores for rules that fire at these positions. (Update scoring details given in text.)

Figure 1: Training Algorithm

iments. Figure 1. omits the specifics of how rule scores are incrementally updated, as this differs between appending and prepending approaches (we return to this matter shortly). An 'efficiency feature' of the algorithm is that in computing the initial scores for TRs, we firstly identify the rules that effect at least one correction somewhere, and then subsequently only score negative changes for these rules. For the appending version, these initially identified rules are the only ones that need be considered during the phase. The situation is similar for the prepending version, except that we must additionally allow for identity rules to be considered as possible rules during incremental update.

Some further efficiencies that are not described in Figure 1. involve using the prespecified stopping threshold for training to prefilter work. When the corpus is first loaded, we can extract counts for all 'error pairs' (x, y) where the default tag x should have been y. We need only entertain TRs changing x to y if the count for the pair (x, y) is above threshold, i.e. since it is otherwise impossible that such TRs could score above threshold. A position with initial tag x is considered a 'data point' in training only if it has an alternative tag y for an above threshold error pair (x, y). Similarly, when we identify the data points for a phase of training, we can count the occurrences of words appearing within the 'context window' around these points. We need then only consider instantiations of lexical templates using words whose appearance count in these contexts is above threshold.

As noted above, the details of the incremental update part of the training algorithm differ for the ICA and ICP variants. For ICA, the scores hash should be viewed as recording, at each stage, the scores for each rule if it were adopted to 'cover' some portion of the remaining training instances. When some new rule T is adopted, we then identify the data points where T fires. These positions now 'belong' to T, and are discounted for subsequent scoring (i.e. deleted from the record of 'data points'), and we must also 'undo' or cancel any scores in the rule scores hash that depend on these positions, i.e. scoring -1 if the rule effects a correction at one of these positions, or $+1$ if it introduces an error. For ICP, the scores hash records, at each stage, the scores for each rule if it were adopted to *override* the existing rule set. Again, when a rule T is adopted, we can update the scores hash by considering only the data points where T fires, but in this case the change made to the score of a possible rule P that could fire at these positions depends not on whether P outputs the correct tag or not, but rather on whether the position would be correct under the previous rule set, and whether it will be correct given T's adoption. Specifically, if a position was correct (resp. incorrect) and T makes it correct (resp. correct), then for rule P that can fire at this position we score $+1$ (resp. -1).[2]

6 The Tagging Algorithm

For Brill's method, the distance at which two rule uses may *directly* affect each other is limited by the width of contexts in rule templates. By chaining, indirect interactions can in theory span over much greater distances. Hence, a tagging algorithm which applies the rules directly cannot work with a narrow window over the text, but instead must work with text units of at least sentence size.

For the present approach, the independence assumption tells us to ignore rule interactions, so we can use an algorithm which streams the text through a buffer that is just wide for single rules to fire. With contexts ranging upto three elements either side, a window width of seven elements suffices. At each step of advancing the window, we examine the initial tag assigned to the central element, access the subclass of rules that can modify this tag, and check these rules in sequence to see if any can apply. If any one (or none) of them fires, we have finished at this position.[3]

7 Evaluating rule-based tagging without rule interaction

We compared the performance of Brill's implementation (for which the code is publicly available) with an implementation in C of the algorithms sketched above, employing the same template set (with both lexical and non-lexical templates), and making the closed vocabulary assumption. A cross-validation approach was used. We took the even numbered sections of the tagged *Wall Street Journal* component of the PTB (comprising 680K tokens) and randomly distributed the sentences into eight bins, resulting in bins of size 85±2K tokens. These collections allow for an eight-fold cross-validation of results, i.e. each training task is repeated eight times, with one of the bins selected as the test data, and the other seven combined to form the training data, giving training set sizes of around 595±2K tokens (comparable to the 600K used by Brill in his larger experiments). The results obtained are averaged to give a more reliable indication of performance than could be gained from any single run.[4]

We trained the Brill system at thresholds 15 (the value suggested for this much training data in the code's instructions) and 12. The results are shown in the Table 1. (where the scores for accuracy, etc, are averages over the eight runs). The final column of the table shows the time taken to tag 1 million tokens using the rule sets produced (again averaged). The table also shows corresponding results for the ICA and ICP training methods, at a range of different training thresholds.

Let us write "method/threshold" (e.g. Brill/15) to refer to a method trained at a given threshold. Observe that the tagging accuracy for ICA/15 is lower, by 0.16%, than for Brill/15. However, ICA here produces less rules than Brill (154 vs. 183), so it is interesting instead to compare ICA/12 and Brill/15, which produce similar rule counts. Here the accuracy of ICA is lower by only 0.1%. Since ICA is the 'minimal variant' of the original Brill method, these results can be viewed as providing direct empirical evidence of the im-

[2]This can be seen by reasoning through the different cases. For example, if the position was previously correct, T makes it incorrect, and P assigns the correct tag, then P's score is incremented, since before T's adoption, P's effect was neutral at this position, and now it will effect a correction. If P instead assigns an incorrect tag, its score is still incremented, since P's effect was previously negative at this position and now its effect will be neutral (i.e. T takes the blame for making the position wrong).

[3]There is a question as to whether we should actually overwrite a tag change into the buffer (rather than just outputting it), so that this is the tag present when rules applying to the next few positions are checked. According to the independence assumption, this decision should matter little, and our informal explorations bear this idea out.

[4]The Brill training experiments were performed on a Sun Enterprise 450 Server, having two 400Mhz UltraSPARC CPUs and >1G of memory. The ICA/ICP training experiments used the same machine, but by the time they were run, it had been ungraded to have four 400Mhz UltraSPARC CPUs.

method	threshold	tagging accuracy (%)	rule count	training: runtime (mins)	memory	execution: runtime (secs) on 10^6 tokens
Brill	15	96.99	183	1310	41	180
Brill	12	97.05	214	1625	41	207
ICA	15	96.83	154	1.6	18	10.5
ICA	12	96.89	182	1.6	18	10.9
ICA	9	96.96	231	1.7	18	11.4
ICA	6	97.03	324	1.9	18	12.3
ICA	4	97.07	459	2.2	18	13.9
ICA	3	97.09	605	2.4	18	15.0
ICA	2	97.11	919	2.7	18	18.4
ICP	15	96.96	175	1.9	19	11.0
ICP	12	97.03	211	2.0	19	11.3
ICP	9	97.12	272	2.1	20	12.0
ICP	6	97.21	389	2.5	21	13.1
ICP	4	97.27	566	2.9	22	15.1
ICP	3	97.30	755	3.3	22	17.1
ICP	2	97.35	1183	4.1	23	21.9

Table 1: Experimental Results (cross-validation, with training sets of ~595K tokens)

portance of rule interaction to Brill's rule-based tagging method, suggesting that it does have an impact on performance, but one that is rather limited. The short training times for ICA allow that we might seek to recoup this loss in accuracy, as compared to Brill, by training at lower thresholds. The accuracy of Brill/15 is nearly matched by ICA/9 and slightly surpassed by ICA/6. The results for ICP bear out the suggestion that prepending might improve performance. The results for ICP/15 and ICP/12 are only very slightly below those for Brill/15 and Brill/12. Training at lower thresholds, ICP produces results that significantly improve on those for Brill/12, but still with very short training times.

A striking feature of the results in Table 1. is the relative insensitivity of ICA/ICP tagging time to rule set size. Comparing ICA/15 and ICA/3, for example, we see a roughly four-fold increase in rule count, but an increase in tagging time of around 40%. By contrast, Brill's method shows an increase in tagging time that is roughly proportional to the increase in rule count. This is not surprising since the tagging algorithm requires a pass over the loaded input for each rule in the sequence. For decision list tagging, only a single pass over the input is made. At each token only a subset of the rules are accessed, and if any rule fires, the remainder are ignored. We might predict slightly slower tagging for ICP rules sets than for ICA, i.e. since the highly general rules that fire most often will appear later in the decision list for ICP than for ICA, and there is some evidence for this in the results (e.g. compare ICP/4 to ICA/3).

8 Discussion

We have shown how Brill's rule-based tagging approach can be reformulated as a decision list model, allowing for much faster training, to give results that are comparable to, or better than, those of Brill's original system. The modified approach allows for tagger execution that is quite fast, even for an implementation performing direct rule interpretation. This implementation, however, probably cannot compare with the speed of a tagger for the standard Brill approach produced using the finite-state compilation method of Roche and Schabes (1995), being run on current hardware. We expect that similar finite-state compilation is possible for the decision list approach, and speculate that, since long-distance interaction effects are excluded, it should be possible to do a compilation producing much smaller finite-state transducers.

A final point to consider is whether examples for which the Brill approach relies on interaction can be correctly handled in the modified approach. We cannot give a general answer to this question, but the following example gives some indication of what may be happening in many cases. The Brill tagger assigns the phrase *to sign up* the initial tags `to/TO sign/NN up/RB`, which are (correctly) changed to be `to/TO sign/VB up/RP` under the rules "`NN` becomes `VB` if previous tag is `TO`" and "`RB` becomes `RP` if word is `up` and previous tag is `VB`", where the second relies on the first having fired. Both ICA and ICP yield rules that correctly handle the case also. For ICP, for example, we have also the first of the Brill rules, but the second rule involved is instead "`RB` becomes `RP` if word is `up` and tag before last is `TO`", which correctly fires without relying on the first rule.

References

Eric Brill. 1993. *A corpus-based approach to language learning.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, December.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Michael Perkowitz. 1993. Equations for part of speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789.

Kenneth Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing.*

James Cussens. 1997. Part-of-speech tagging using Progol. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 93–108.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 133–140.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. Mbt: A memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27.

Torbjörn Lager. 1999. The μ-TBL system: Logic programming tools for transformation-based learning. In *Proceedings of the Third International Workshop on Computational Natural Language Learning (CoNLL'99)*, Bergen.

Lance A. Ramshaw and Mitchell P. Marcus. 1994. Exploring the statistical derivation of transformation rule sequences for part-of-speech tagging. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 86–95.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

Ronald L. Rivest. 1987. Learning decision lists. *Machine Learning*, 2(3):229–246.

Emmanuel Roche and Yves Schabes. 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2):227–253.

Ken Samuel. 1998. Lazy transformation-based learning. In *Proceedings of the Eleventh International Forida AI Research Symposium Conference*, pages 235–239.

Hans van Halteren, Jakub Zavrel, and Walter Daelemens. 1998. Improving data-driven word-class tagging by system combination. In *Proceedings of the International Conference on Computational Linguistics COLING-98*, pages 491–497.

Geoff Webb and N. Brkic. 1993. Learning decision lists by prepending inferred rules. In *Proceedings of the AI–93 Workshop on Machine Learning and Hybrid Systems*, pages 6–10, Melbourne.