Part-of-Speech (POS) Tagging Revisited

Mark Sharp MSHARP@SCILS.RUTGERS.EDU
School of Communication, Information and Library Studies, Rutgers University, 4 Huntington Street, New Brunswick, NJ 08901 USA

Abstract

Accurate part-of-speech (POS) tagging of natural language text data can add power to automated information retrieval and extraction. Brill's transformation-based learning (TBL) approach to automated POS tagging was introduced in 1992, combining virtues of rule-based and stochastic methods. Brill's innovative idea was to use machine learning techniques to search through all of rule space for the most effective rules for tagging ambiguous tokens based on their contexts, but this is computationally infeasible. Typical heuristic patches used in Brill tagging to reduce rule space to tractable sizes seem arbitrary, unempirical, and weak. Now faster hardware and powerful public domain machine learning software offer an opportunity to revisit POS tagging with improved tools. This paper examines TBL and "shotgun" POS tagging using the Weka J48 classifier on text corpora from different genre domains.

1. Introduction

Classical token-based, bag-of-words automated information retrieval (IR) and information extraction (IE) methods appear to have "hit a wall" performance-wise. Many information science researchers see this as a sign that social or "user-oriented" factors must be taken into account in IR/IE system design. Machine-oriented traditionalists, however, point to recent advances in hardware, algorithms, and knowledge bases which have reinvigorated interest in natural language processing (NLP) and computational linguistics approaches to IR and IE. The evidence on both sides is inconclusive and the debate is ongoing.

Accurate part-of-speech (POS) tagging is the critical first step of NLP for IR and IE (Vasilakopoulos, 2003). Brill (1992) introduced a transformation-based learning (TBL) approach to automated POS tagging which combined the linguistic intuition of traditional rule-based approaches with the empirical practicality of traditional stochastic methods and which remains a popular method (Jurafsky & Martin, 2000).

However, Brill's fundamental concept of a complete search through "rule space" followed by empirical decision making has never been implemented because it is computationally infeasible for any text corpus of meaningful size. That is, one cannot correlate the assignment of one or another POS tag to a given word token with every possible combination of other tokens and tags in the text since the latter is virtually infinite. Brill worked around this problem by limiting attribute space to the POS tags within ±3 word positions of the target token and limiting the rule syntax to a few "templates," but this seems drastic and arbitrary. Subsequent advances in hardware speed and machine learning algorithms suggest that some of Brill's criticisms of pure stochastic methods may be outdated, and that automated POS tagging might now be fruitfully revisited.

2. Prior Research

The Weka project has made publicly available a library of powerful automated classifiers implemented in Java (Witten & Frank, 1999). Although widely used to teach and practice machine learning, Weka has apparently not been used for POS tagging *per se*. Finn, Kushmerick, and Smyth (2002) used Weka J48 (a variant of the C4.5 pruned decision tree algorithm) to classify WWW pages into two classes (fact and opinion) by POS statistics, which they obtained from the output of a Brill tagger -- i.e., a "bag of POS tags" vector, one percentage for each of 36 POS features. Vasilakopoulos (2003) used J48 to infer the initial POS tag for "unseen" words (words that occur in the test set but not the training set) based on eight attributes of the unseen word and its context. O'Hara and Wiebe (2003) used J48 to disambiguate prepositions based on the POS tags at *n-2, n-1, n, n+1,* and *n+2* word positions as attributes. This is very close to the current design, the difference being that here we want to predict transformations of the initial POS tag at position *n*.

Prior work (*brill2.txt*[1]) showed that the conveniently small SUSANNE Corpus[2] could be adapted for use in TBL-

[1] This and other scripts and files so noted may be downloaded from http://scils.rutgers.edu/~msharp/CS536_ML/.

[2] ftp://crl.nmsu.edu/CLR/lexica/Susanne/

POS tagging experiments. The corpus consists of 16 texts from the Brown Corpus drawn from press reports, belles lettres, biographies, memoirs, learned (mainly scientific and technical) writing, and adventure and Western fiction. The SUSANNE POS tagset numbers a cumbersome 358, but these can be converted to the standard Penn Treebank[3] set of 45 plus a catch-all for the uncertain ones (UNC). The cleaned, tagged corpus contains 48257 tokens of which 571 are unique and have more than one Penn Treebank POS tag. The total frequency of the subordinate (minority) tag on these tokens is 1966, or about 4.1% of the corpus. That is, the baseline error rate of POS tagging by dominant (majority) tag for each word with no transformation is about 4.1%. By far the largest single contributor is the word "to" used as a preposition (tag=IN) with 391 occurrences compared to the infinitive verb auxiliary "to" (tag=TO) with 579 occurrences. The baseline learned tagging rule for "to" is thus:

if (token[n] EQUALS "to")

then set tag[n] to "TO"

where *n* signifies the word position in the text. The following transformation was devised by ad hoc manual analysis:

if (tag[n+1] MEMBER_OF {$: CD DT JJ JJR NN NNP NNS PDT PP PP$ VBG VBN WDT WRB})

then change tag[n] to "IN"

This transformation corrects 381 of the 391 TO/IN errors while introducing only 2 new errors by changing correct TO tags to IN, thus reducing the error rate by 379 tags or about 0.8% to 3.3%.

3. Experimental Methods and Results

3.1 Shotgun Approach

A perl script (*Weka571_42.p*) was written to extract the multiply tagged tokens, their supervised tags, and their *n±10* token-tag contexts from a working version of SUSANNE (*susanne3.txt*) and output the data into an "arff" file for Weka. Tokens with non-alphabetic characters were conflated to "nonalph" and tags with non-alphabetic characters were "alpha-ized" (e.g., "."="PER") to expedite Weka compatibility. The arff file contained 42 attributes (21 tokens and 21 tags at *n-10, n-9,…n…n+9, n+10*) and 23710 tuples. Attempting to run Weka J48 on it resulted in an "OutOfMemoryError" on a Pentium IV/256 Mb RAM PC running Windows XP. Reducing the attribute dimensionality to 10 (*n-2, n-1, n, n+1, n+2*) still resulted in an "OutOfMemoryError". Thus we are reminded that Brill's critique of the training data-intensiveness of stochastic methods is not just about obtaining the data but also about the computational feasibility of processing it.

However, it was feasible to process all 42 attributes for the single target token "to" at *n* (*Weka1_42a.p*) for which we had the comparison data discussed above. J48 constructed a decision tree equivalent to the following rule:

if (token[n] EQUALS "to")

then set tag[n] to "TO"

if (tag[n+1] MEMBER_OF {CD ***COL DOL*** *DT JJ JJR NN NNP NNS PDT PP* ***PPS*** *RB VBG VBN WDT WRB}* **[6 new cases]**

then change tag[n] to "IN"

if (tag[n+1] EQUALS "RB")

if (tag[n+2] EQUALS "VB") **[3 new cases]**

then change tag[n] back to "TO"

The differences with the ad hoc rule devised by hand are bolded. COL, DOL, and PPS are just artifacts of "alpha-ization" of the tags. Dealing with RB, however, adds real new power to the classifier. In ad hoc rule, we chose to exclude the *n+1* tag value RB because it created 3 new errors while correcting only 3 old errors. J48 "figured out" how to correct the 3 new errors by discriminating on the *n+2* tag and changing the *n* tag back to "TO" if the *n+2* tag was VB. This makes sense, since RB=adverb and VB=verb, the sequence "to"-RB-VB is likely to be an infinitive verb phrase such as "to boldly go". It is interesting that the post-pruning built into the C4.5 process whittled down the 40 optional attributes to just two, both of which were tags on tokens within *n±3*, giving the decision tree the look of one of Brill's heuristic templates (Jurafsky & Martin, 2000, p. 312). The training set error was 0.72% (7 errors out of 970 cases) and the 10-fold cross-validation error was 1.13% (11 errors).

The computational infeasibility of this design for larger numbers of target tokens arises from attempting to use the surrounding tokens as attribute values. In contrast to the tags, which may only take on 46 different values at most, the tokens may take on the thousands of values found within the context window of all occurrences of all the unique target tokens. Using the token values is desirable and was a key objective of this project, since it would contrast sharply with the usual implementation of Brill taggers which rely on the tags alone.

Eliminating the token context attributes yielded a design with 571 target tokens and 22 attributes (target token at *n*, POS tag class at *n*, *n±10* POS tags) that J48 could handle (*Weka571_22.p*). The resulting pruned tree (*susanne571_22tree.txt*) had over 4000 nodes equivalent to 317 transformation rules given prior dominant tagging (*susanne571_22rules.txt*). The training set error was 7.1% (1044/14674) and the 10-fold cross-validation error

[3] http://www.cis.upenn.edu/~treebank/home.html.

was 11.3% (1657/14674). This means that the overall training error rate would be reduced from the baseline of 4.1% to 2.2% (1044/48258) by the single transformation rule represented by this tree, a significant improvement over the ad hoc "to" rule in both accuracy gain and ease of discovery. The contrast in complexity (and therefore similarity to Brill-like simple heuristics) of the two rules is striking. The J48 tree uses the *n* token values at the top level, then goes up to three nested splits deep making decisions based on the context tags from *n-9* to *n+8*, a far wider net than that cast by Brill's *n±3* templates.

3.2 TBL Approach

TBL can be likened to creating a painting by overlaying the appropriate colors in decreasing order of the area they occupy in the final picture (Jurafsky & Martin, 2000, p. 307). The 571x22 shotgun method described above is more like throwing all the paint at the canvas at once. Can it be improved by dealing with one target token at a time rather than all 571 at once? Might such an approach allow the token values back into feasible attribute space?

An adaptation of the "to" script (*Weka1_42b.p*) was created to extract from a secondary SUSANNE file (*susanne4.txt*) the *learned* rather than *supervised* tag values for the *n±10* attribute values. The token values and supervised class tag value remained the same. The resulting J48 tree was simpler, only one level deep (no splits), equivalent to the following rule:

if (token[n] EQUALS "to")

then set tag[n] to "TO"

if (tag[n+1] MEMBER_OF
{CD COL DOL DT ***IN*** *JJ JJR NN NNP NNS*
PDT PP PPS RB VBG VBN WDT WRB}

then change tag[n] to "IN"

The differences with the prior result are the *IN* value and the loss of the RB error correction. Since there are 1966 errors in the untransformed dominant learned tags, equivalent to a 4.1% noise level, it is not surprising that there J48 training error rate for predicting the TO/IN transformation on "to" is higher: 5.26% compared to 0.72% using the supervised tags.

If some of the ambiguous tokens are within each others' context windows, transformation of their learned tags would interact if done sequentially. One might expect the learned-supervised error discrepancy for each transformed token to decrease as more tokens are transformed earlier, thereby increasing the accuracy of the learned tags. This suggests that the tokens should be processed in order of how many errors they contribute to the baseline learned tag set in order to maximize overall accuracy -- i.e., "to" should be done last, not first.

A truly integrated system would combine the functionality of J48, the arff-creating code, and more code (*POSrule1.p*) which transforms the learned tag values in SUSANNE based on the rule learned by J48. Such a system could be made to cycle iteratively through all 571 target tokens starting with those that only contribute one error and working up to "to" which contributes 391. For proof of concept, I manually simulated such a process for the 17 of the top 40 error-contributing tokens. These 17 tokens contributed 690 of the 1966 dominant tag errors and gave a productive J48 decision tree when analyzed individually in order of error counts (*TBLrules.txt*). The other 23 of the top 40 had higher dominant tag accuracy and so J48 did not generate a transformation rule.

The results of this simulation did not support either hypothesis stated above. The overall error rate was 1363/48257 ~ 2.8%, greater than the 2.2% achieved by the simpler shotgun method, and the individual error rate for "to" was 5.36%, essentially equal to the 5.26% achieved by learning the "to" transformation rule first. Subsequently it was determined that in the simulation only one of the 260 prior transformations was within the context window of the final "to" rule (*findtokens2.p*). Adding a shotgun J48 analysis of the other 554 (=571-17) multi-tagged tokens did not help much, since the resulting rule only corrected 333 of their 1276 dominant tag errors. If added to the performance of the 17 sequential rules, this reduces the overall error rate to 1030/48257 ~ 2.1%, essentially equal to the 2.2% of the shotgun method. These results are shown graphically in Figure 1.

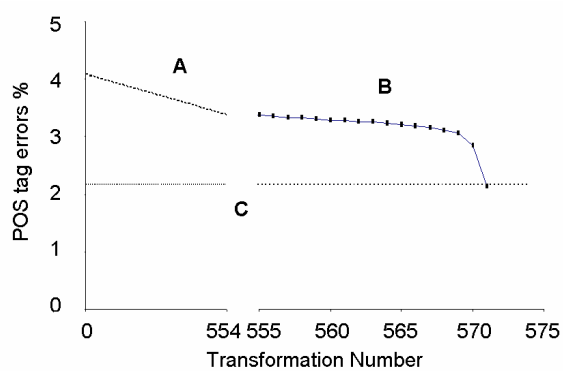

Figure 1. Simulated TBL POS tagging of the Susanne corpus using J48 trees as token-specific rules. **A**: shotgun rule for 554 low impact tokens. **B**: individual rules for 17 high impact tokens applied in order of impact. **C**: shotgun rule for all 571 tokens (non-TBL benchmark for comparison).

3.3 Cross-domain robustness

The 571x22 shotgun rule (curve C in Figure 1) was tested on two larger POS-tagged corpora. The first corpus was derived from the 2002 Penn Treebank Wall Street Journal (WSJ) collection sets 00-10 concatenated, formatted, and

cleaned of ambiguous tags (*wsjconvert.p*, *wsjcleanup.xls*) yielding a corpus of 555712 tokens (*wsj1.txt*) with a baseline dominant tag error rate (*tagger2.p*) of 5.3%. The second test corpus was derived from a cleaned version of GENIA[4] 3.0 (Sharp & Liu, 2003), a collection of Medline abstracts, with 521164 tokens and a baseline error rate of 2.7%. Both of these test corpora represent different genre domains than SUSANNE so cross-testing should provide a strong test of tagger robustness. Perl scripts were written to apply the transformation rule derived from the SUSANNE 571x22 pruned J48 tree to either the supervised (*tagger3sup.p*) or learned/dominant tags (*tagger3lea.p*). The results are shown in Table 1.

Table 1. POS tagging error rates pre and post transformation by the SUSANNE 571x22 shotgun rule using (a) supervised tags, (b) learned/dominant tags.

	Corpus	#tokens	%trans formed	Pre %err	Post %err
a	SUSANNE	48257	1.7	4.1	2.5
a	WSJ	555712	1.2	5.3	6.1
a	GENIA	521164	1.1	2.7	3.7
b	SUSANNE	48257	1.8	4.1	2.7
b	WSJ	555712	1.4	5.3	6.4
b	GENIA	521164	1.1	2.7	3.7

The transformation actually increased the error rate in both test corpora. Subsequently it was discovered that "to" is almost never tagged as IN in WSJ or GENIA (=0.1% vs. 41% in SUSANNE). When the "to" part of the transformation was eliminated the post-transformation error rates dropped to 5.2% for WSJ and 2.8% for GENIA, roughly equal to the pre-transformation rates. A TBL simulation was compared to a shotgun rule for 46 of the top 50 error-contributing tokens in GENIA, and again the correction rate was virtually the same (3999 vs. 4000 out of 4629 errors corrected) due to lack of overlap of the token context windows (*geniacounts.xls*).

Although the control run of the transformation scripts on SUSANNE reduced the error rate, it was only to 2.5%, not the 2.2% given by the J48 output. This discrepancy persisted despite diverse methods of deriving the perl code from the J48 tree. It suggests that the training error calculated by J48 may not be actually attainable by implementing the learned rule due to the interdependence of the tags on neighboring low-error tokens. This insight puts the actual performance of the TBL approach (curves A and B in Figure 1) in a better light (that is, if curve C is raised to 2.5%). J48 was incapable of learning a comparable tree from either WSJ or GENIA with available memory despite several applications of dimensionality reduction, so the reverse cross-testing could not be done.

[4] http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/index.html

4. Discussion

While many nuances could not be explored due to time constraints, in general the results of this study provided a surprising amount of support for Brill's approach to POS tagging. The intuitive appeal of the shotgun approach of dumping dozens or hundreds of context attributes into a Weka learner and computing "one big rule" for all the ambiguous tokens at once proved to be infeasible in practice (due to memory limitations) and possibly in principle (due to the potential interdependence of the tags on neighboring tokens). Simulations of a J48-based, one-token-at-a-time TBL approach showed no improvement over the shotgun method, but both approaches could be better tested using bigger hardware and an integrated, iterative TBL program. For the single-token TBL rules, the general tightness of the context windows and tag (not token) decision attributes computed by the J48 pruning process empirically validate Brill's templates from an objective information-theoretic perspective. While other classifiers such as artificial neural networks, genetic algorithms, and support vector machines might be tested as well, J48 decision trees are especially well-suited to POS tagging due to their facile interpretability as linguistic rules.

References

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Third ACL Applied NLP*, Trento, Italy, 152-155.

Finn, A., Kushmerick, N., & Smyth, B. (2002). Genre classification and domain transfer for information filtering. In *Proceedings of ECIR-02, 24th European Colloquium on Information Retrieval Research.*

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Upper Saddle River, NJ: Prentice-Hall.

O'Hara, T., & Wiebe, J. (2003). Preposition semantic classification via TREEBANK and FRAMENET. Online: http://www.cs.nmsu.edu/~aisem/spring03/ohara-preposition-classification-acl03.pdf

Sharp, M., & Liu, L. (2003). CS533 Term Project: NLP experiments with GENIA. Online: http://scils.rutgers.edu/~msharp/CS533_NLP/CS533final.doc

Vasilakopoulos, A. (2003) Improved unknown word guessing by decision tree induction for POS tagging with TBL. In: *Proceedings of CLUK 2003*, Edinburgh 2003.

Witten, I. H., & Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Francisco: Morgan Kaufmann.