Beáta Megyesi
Department of Linguistics
University of Stockholm
Extract from D-level thesis (section 3)

Brill's rule-based PoS tagger

Beáta Megyesi

Eric Brill introduced a PoS tagger in 1992 that was based on rules, or transformations as he calls them, where the grammar is induced directly from the training corpus without human intervention or expert knowledge. The only additional component necessary is a small, manually and correctly annotated corpus - the training corpus - which serves as input to the tagger. The system is then able to derive lexical/morphological and contextual information from the training corpus and 'learns' how to deduce the most likely part of speech tag for a word. Once the training is completed, the tagger can be used to annotate new, unannotated corpora based on the tagset of the training corpus.

The rule-based part of speech tagger can be said to be a hybrid approach, because it first uses statistical techniques to extract information from the training corpus and then uses a program to automatically learn rules which reduces the faults that would be introduced by statistical mistakes (Brill, 1992). The tagger does not use hand-crafted rules or prespecified language information, nor does the tagger use external lexicons or lists of different types. According to Brill (1992) 'there is a very small amount of general linguistic knowledge built into the system, but no language-specific knowledge'.

The long time goal of the tagger is, in Brill's own words, to 'create a system which would enable somebody to take a large text in a language he does not know and with only a few hours of help from a speaker of the language accurately annotate the text with part of speech information.' (Brill & Marcus, 1992b:1). For achieving his aim, Brill has also developed a parser, consisting of systems which may automatically derive word classes and the bracketing structure of sentences, assigning nonterminal labels to the bracketing structure and improving prepositional phrase attachment (see Brill, 1992, 1993a, 1993b, 1995b).

In this work the part of speech tagger, i.e. learning the most likely tag for a word will be presented. The following section gives a description of the main ideas of Brill's tagger and some information about how to train and test that tagger. All information contained in the following section is based on Brill's articles, listed in the reference list (Brill, 1992; 1993a; 1993b; 1993c; 1994a; 1995a; 1995b). The description below is only functional, disregarding any efficiency aspects considered in the actual software.

Transformation-based error-driven learning

The general framework of Brill's corpus-based learning is so-called Transformation-based Error-driven Learning (TEL). The name reflects the fact that the tagger is based on transformations or rules, and learns by detecting errors. First, a general description of how TEL works in principle is given, then a more detailed explanation for the specific modules of the tagger will follow in subsequent sections.

Roughly, the TEL, shown in Figure 1, begins with an unannotated text as input which passes through *the initial state annotator*. It assigns tags to the input in some fashion. The output of the initial state annotator is a *temporary corpus* which is then compared to a *goal corpus* which has

been manually tagged. For each time the temporary corpus is passed through the *learner*, the learner produces one new rule, the single rule that improves the annotation the most (compared with the goal corpus), and replaces the temporary corpus with the analysis that results when this rule is applied to it. By this process the learner produces an ordered list of rules.

The tagger uses TEL twice: once in the lexical module deriving rules for tagging unknown words, and once in the contextual module for deriving rules that improve the accuracy. Both modules use two types of corpora: the goal corpus, derived from a manually annotated corpus, and a temporary corpus whose tags are improved step by step to resemble the goal corpus more and more.

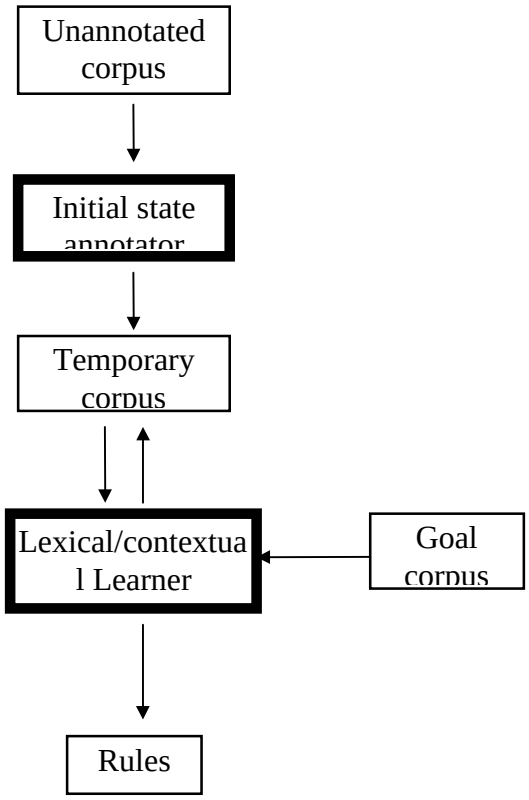

Figure 1. Error-driven learning module (data marked by thin lines)

In the lexical module of the tagger the goal corpus is a list of words containing information about the frequencies of tags in a manually annotated corpus. A temporary corpus on the other hand is a list consisting of the same words as in the goal corpus, tagged in some fashion. In the contextual learning module the goal corpus is a manually annotated running text while a temporary corpus consists of the same running text as in the goal corpus but with different tags.

A rule consists of two parts: a condition (the trigger and possibly a *current tag*), and a *resulting tag*. The rules are instantiated from a set of predefined transformation templates. They contain uninstantiated variables and are of the form

If Trigger, then change the tag X to the tag Y

or

If Trigger, then change the tag to the tag Y

where X and Y are variables. The interpretation of the first type of the transformation template is that if the rule triggers on a word with *current tag* X then the rule replaces *current tag* with *resulting tag* Y. The second one means that if the rule triggers on a word (regardless of the current tag) then the rule tags this word with *resulting tag* Y. The set of all permissable rules PR are generated from all possible instantiations of all predefined templates.

The rule generating process takes an initially tagged[1] temporary corpus TC_0 and finds the rule in PR which gets the best score when applied to TC_0 (a high score for a rule means that the temporary corpus produced when applying the rule gives an annotation closer to the goal corpus). Call this rule R1. Then R1 is applied to TC_0, producing TC_1. The process is now repeated with TC_1, i.e. it finds the rule in PR which gets the best score when applied to TC_1. This will be rule R2[2] which then is applied to TC_1 producing TC_2. The process is reiterated producing rules R3, R4, ... and corresponding temporary corpora TC_3, TC_4, ... until the score of the best rule fails to reach above some predetermined threshold value. The sequence of temporary corpora can be thought of successive improvements closer and closer to the goal corpus. Note that only the tags differ in the different temporary corpora. The output of the process is the ordered list of rules R1, R2, ... which will serve to tag new unannotated texts.

The score for a rule R in PR is computed as follows. For each tagged word in TC_i the rule R gets a score for that word by comparing the change from the current tag to the resulting tag with the corresponding tag of the word in the goal corpus. How this is computed depends on the module. A positive score means that the rule improves the tagging of this word, and a negative score means that the rule worsens the tagging. If the condition of the rule is not satisfied then the score is zero. The total score for R, score(R), is then obtained by adding the scores for each word in TC_i for that rule. When the total score for each R is obtained the rule which has the highest score is added to the set of rules which have alrady been learned. Rules are ordered, i.e. the last rule is dependent on the outcome of earlier rules.

The following sections present an overview of the two modules in Brill's system: the lexical module and the contextual module. First, the lexical module uses transformation-based error-driven learning to produce lexical tagging rules. Then the contextual module, also using transformation-based error-driven learning, produces contextual tagging rules. When using these rules to tag an unannotated text one first applies an initial state annotator to the unknown words (i.e. words not being in the lexicon), then applies the ordered lexical rules to these words. The known words are then tagged with the most likely tag and finally the ordered contextual rules are applied to all words. The lexical and the contextual modules have different transformation templates. The triggers in the lexical module depend on the character(s) and the affixes of a word and some of them depend on the following/preceding word. The triggers in the contextual module, on the other hand, depend on the current word itself, the tags or the words in the context of the current word.

[1] Initial state annotation is done differently depending on which learning module is used. This will be explained in detail later.

[2] Note that R2 could in principle be the same rule as R1, i.e. the rules considered in each iteration is the *whole* set PR.

Learning lexical rules

The ideal goal of the lexical module is to find rules that can produce the most likely tag for any word in the given language, i.e. the most frequent tag for the word in question considering all texts in that language. The problem is to determine the most likely tags for unknown words, given the most likely tag for each word in a comparatively small set of words.

The lexical learner module is weakly statistical. It uses the first half of the manually tagged corpus[3] as well as any additional large unannotated corpus containing the manually tagged corpus with the tags removed. The learner module uses three different lexicons or lists constructed (by the user) from the manually tagged corpus and the large unannotated corpus. The `smallwordtaglist`, built from the manually annotated corpus, serves as the *goal corpus* in TEL and contains lines of the form

Word Tag Frequency.

It simply contains the frequencies of each `Word/Tag` pair in the manually annotated corpus. Freq(W,T) denotes the `Frequency` of a word W with a specific tag T, and Freq(W) denotes the frequency of the word `Word` in the manually annotated corpus. These numbers are used by the lexical learner in two ways: Freq(W,T) is used to compute the most likely tag T for the word W (i.e. the one with the highest frequency), and

P(T|W) := Freq(W,T) / Freq(W)

is the estimated probability that the word W is tagged with the tag T.

The other two lists are called `bigwordlist` and `bigbigramlist` and are constructed from the large unannotated corpus. `bigwordlist` is a list of all words occurring in the unannotated corpus, sorted by decreasing frequency. The `bigbigramlist` is a list consisting of all word pairs (hence the name bigram) occurring in the unannotated corpus. `Bigbigramlist` does not contain the frequencies of word pairs; it only records if a given word pair occurs in the unannotated corpus or not. Note that once the user has constructed these lists, the lexical learner does not need either the manually annotated or the unannotated corpus because the lists contain all the information the learner needs. The `bigwordlist` and the `bigbigramlist` are only used to check the trigger condition, which is completely determined by the data in these two lists.

First, the learner constructs a word list from `smallwordtaglist`, i.e. the words with tagging information removed. Then the initial state annotator assigns to every word the default most likely tag. The default tags for English are NN and NNP[4], where NNP is assigned to words which start with a capital letter and NN is assigned to words which do not start with a capital letter. The word list thus obtained is the initial *temporary corpus* WL_0; it was called TC_0 in the general description of TEL above.

Once WL_0 has been produced by the initial state annotator the learner generates the set of all permissable rules PR from all possible instantiations of all the predefined lexical templates and computes a score for every rule R in PR (see below). The rule which achieves the best score becomes rule number one in the output. Then the learner transforms WL_0 to WL_1 by applying this

[3] The second half of the manually annotated corpus will be used with the contextual rule learner.

[4] In the case of the Hungarian corpus I simply used these default tags as nonterminal tags.

rule. This process is repeated until no rule can be found with a score greater than some threshold value, i.e. compute the new scores for all the rules in PR, pick the one with the best score, output this rule as rule number two and apply it on WL_1 to get WL_2, etc.

The scoring function is defined as follows: If the rule R has the template:

if Trigger then change tag X to tag Y

and *w* is a word in WL_i with current tag X satisfying the trigger condition, then R gets the score P(Y|*w*) - P(X|*w*) for the word *w*. The total score for R is then obtained by adding all the 'word scores'.

$$\text{score}(R) := \sum_{w} P(Y|w) - P(X|w)$$

where the sum runs through all *w* in WL_i with current tag X, satisfying the trigger condition.

If the rule R has the template:

if Trigger then change current tag to tag Y

and *w* is a word in WL_i satisfying the trigger condition, then R gets the score P(Y|*w*) - P(Current tag of *w* |*w*) for the word *w*. The total score for R is then obtained by adding all the 'word scores'.

$$\text{score}(R) := \sum_{w} P(Y|w) - P(\text{Current tag of } w \,|w)$$

where the sum runs through all *w* in WL_i, satisfying the trigger condition. Note that the score the rule R gets for *w* always is of the form P(new tag|*w*)-P(old tag|*w*). A positive score means that the new tag is more likely than the old tag while a negative score means that the new tag is less likely than the old tag. The trigger condition is tested using `bigwordlist` and `bigbigramlist`, and the estimated probabilities are computed from the frequencies in `smallwordtaglist`.

The set of templates used and the name of the rule type for each template in parenthesis are given below, (see also Appendix B).

1. Change the most likely tag to Y if the character Z appears anywhere in the word. *(char)*

2. Change the most likely tag to Y if the current word has suffix x, $x \leq 4$. *(hassuf)*

3. Change the most likely tag to Y if deleting the suffix x, $|x| \leq 4$, results in a word. *(deletesuf)*

4. Change the most likely tag to Y if adding the suffix x, $|x| \leq 4$, results in a word. *(addsuf)*

5. Change the most likely tag to Y if the current word has prefix x, $x \leq 4$. *(haspref)*

6. Change the most likely tag to Y if deleting the prefix x, |x| ≤ 4, results in a word. *(deletepref)*

7. Change the most likely tag to Y if adding the prefix x, |x| ≤ 4, results in a word. *(addpref)*

8. Change the most likely tag to Y if word W ever appears immediately to the left/right of the word[5]. *(goodleft)* vs. *(goodright)*

9. Change the most likely tag from X to Y if the character Z appears anywhere in the word. *(fchar)*

10. Change the most likely tag from X to Y if the current word has suffix x, x ≤ 4. *(fhassuf)*

11. Change the most likely tag from X to Y if deleting the suffix x, |x| ≤ 4, results in a word. *(fdeletesuf)*

12. Change the most likely tag from X to Y if adding the suffix x, |x| ≤ 4, results in a word. *(faddsuf)*

13. Change the most likely tag from X to Y if the current word has prefix x, x ≤ 4. *(fhaspref)*

14. Change the most likely tag from X to Y if deleting the prefix x, |x| ≤ 4, results in a word. *(fdeletepref)*

15. Change the most likely tag from X to Y if adding the prefix x, |x| ≤ 4, results in a word. *(faddpref)*

16. Change the most likely tag from X to Y if word W ever appears immediately to the left/right of the word. *(fgoodleft)* vs. *(fgoodright)*

Note that the only difference between the first eight and the second eight templates is that in the latter the template changes a tag X to another tag Y while in the first eight templates the new tag will be Y regardless of the current tag.

The lexical rules produced by the lexical learning module are then used to initially tag the unknown words in the contextual training corpus. This will be described in detail below.

Learning contextual rules

Once the tagger has learned the most likely tag for each word found in the manually annotated training corpus and the method for predicting the most likely tag for unknown words, contextual

[5] These templates are costly to compute since for *each* word W we get a template of the simpler type (the templates except number 8 and 16). To speed upp the learning process a threshold n is given as an argument to the learner which restricts the use of these templates to the n most frequent words in `bigwordlist` (i.e. the first n words in `bigwordlist`). This is the reason `bigwordlist` is sorted.

rules are learned for disambiguation. The learner discovers rules on the basis of the particular environments (or the context) of word tokens.

The contextual learning process needs an initially annotated text. The input to the initial state annotator is an untagged corpus, a running text which is the second half of the manually annotated corpus where the tagging information of the words is removed. The initial state annotator needs a list, a so called `traininglexicon`, which consists of list of words with a number of tags attached to each word. These tags are the tags that are found in the *first half* of the manually annotated corpus (the ones used by the lexical module). The first tag is the most likely tag for the word in question and the rest of the tags are in no particular order.

Word Tag_1 Tag_2 … Tag_n

With the help of the `traininglexicon`, the `bigbigramlist` (the same as used in the lexical learning module, se above) and the lexical rules, the initial state annotator assigns to every word being in the untagged corpus the most likely tag. It tags the known words, i.e. the words occurring in `traininglexicon`, with the most frequent tag for the word in question. The tags for the unknown words are computed using the lexical rules: each unknown word is first tagged with NN or NNP and then each of the lexical rules are applied in order. The reason why `bigbigramlist` is necessary as input is that some of the triggers (nr. 8 and 16) in the lexical rules are defined in terms of this list[6]. The annotated text thus obtained is the initial *temporary running text* RT_0[7] which serves as input to the contextual learner.

Transformation-based error-driven learning is used to learn contextual rules in a similar way as in the lexical learner module. The input to the contextual learner is the second half of the manually annotated corpus (i.e. the goal corpus), the initial temporary corpus RT_0 and the `traininglexicon` (the same one as above). First, the learner generates the set of all permissable rules PR from all possible instantiations of all the predefined contextual templates. Note that PR in the contextual module and PR in the lexical module are different sets of rules, since the two modules have different transformation templates. Here, the triggers of the templates for the rules usually depend on the current context. The following are the triggers of the contextual transformation templates:

1. The preceding/following word is tagged with Z. (PREVTAG/NEXTTAG)
2. One of the two preceding/following words is tagged with Z. (PREV1OR2TAG/NEXT1OR2TAG)
3. One of the three preceding/following words is tagged with Z. (PREV1OR2OR3TAG/NEXT1OR2OR3TAG)
4. The preceding word is tagged with Z and the following word is tagged with V. (SURROUNDTAG)
5. The preceding/following two words are tagged with Z and V. (PREVBIGRAM/NEXTBIGRAM)
6. The word two words before/after is tagged with Z. (PREV2TAG/NEXT2TAG)
7. The current word is Z. (CURWD)

[6] These triggers do not check if the word before/after the current word is a particular word (in the current context), instead they check if the current word ever can be preceded/followed by a particular word in some context.

[7] RT_0 was called TC_0 in the general description of TEL above.

8. The preceding/following word is W. (PREVWD/NEXTWD)
9. One of the preceding/following words is W. (PREV1OR2WD/NEXT1OR2WD)
10. The word two words before/after is W. (PREV2WD/NEXT2WD)
11. The current word is Z and the preceding word is V. (LBIGRAM)
12. The current word is V and the following word is Z. (RBIGRAM)
13. The current word is V and the preceding/following word is tagged with Z. (WDPREVTAG/WDNEXTTAG)
14. The current word is V and the word two words before/after is tagged with Z. (WDAND2BFR/WDAND2TAGAFT)

For all rules in PR for which the trigger condition is met the scores on the temporary corpus RT_0 are computed. It picks the rule with the highest score, R1, which is then put on the output list. Then the learner applies R1 to RT_0 and produces RT_1, on which the learning continues. The process is reiterated putting one rule (the one with the highest score in each iteration) on the output list[8] in each step until learning is completed, i.e. no rule achieves a score higher than some predetermined threshold value. A higher threshold speeds up the learning process but reduces the accuracy because it may eliminate effective low frequency rules.

If R is a rule in PR, the score for R on RT_i is computed as follows. For each word in RT_i the learner computes the score for R on this word. Then the scores for all words in RT_i where the rule is applicable are added and the result is the total score for R. The score is easy to compute since the system can compare the tags of words in RT_i with the correct tags in the goal corpus (the texts are the same). If R is applied to the word *w*, thereby correcting an error, then the score for *w* is +1. If instead an error is introduced the score for *w* is -1. In all other cases the score for *w* is 0. Thus, the total score for R is

score(R) := number of errors corrected - number of errors introduced.

There is one difference compared to the lexical learning module, namely the application of the rules is restricted in the following way: if the current word occurs in the `traininglexicon` but the new tag given by the rule is not one of the tags associated to the word in the `traininglexicon`, then the rule does not change the tag of this word.

The rules produced by the contextual learning module together with the lexical rules and several other files can then be used as input to the tagger to tag unannotated text, as will be described in the next section.

Tagging new texts

To execute the tagger several files are used: lexicon, an unannotated corpus which will be tagged by the tagger, bigbigramlist, lexical rule file and contextual rule file. For calling the tagger, the following command has to be written:

tagger LEXICON UNTAGGEDCORPUS BIGBIGRAMLIST LEXICALRULEFILE \ CONTEXTUALRULEFILE > Outputfile

[8] Recall that the output list is ordered.

`bigbigramlist` is the same as above, the lexical and contextual rule files are the ones computed by the lexical and the contextual learner modules. The lexicon is built in the same way as `traininglexicon` above, the only difference being that the whole manually tagged corpus is used. The `bigbigramlist` is needed to check the trigger conditions for the lexical rules.

First, the tagger tags all known words (words occurring in the lexicon) with the most frequently occurring tag for the word in question. Then, the tags for the unknown words are computed using the lexical rules: each unknown word is first tagged with NN or NNP and then each of the lexical rules are applied in order. This completes the lexical part of the tagging. Then, the contextual rules are applied to the whole corpus in order. The result is the annotated text.

Some practical information

The software is distributed freely and is available from

'ftp.cs.jhu.edu/pub/brill/Programs/RULE_BASED_TAGGER'

Information about how to compile the programs and how to train and test the tagger is found in README files in the directory Docs.

Note that all text files, both annotated and unannotated used for training and testing must be tokenised which includes removing the punctuation from words and placing one sentence per line. Tagged text is of the form word/tag. For example, annotated text taken from the Hungarian test corpus has the following form,

De/KOT keme1nyse1ge/FN_PSe3 volt/IGE_Me3 a/DET legnagyobb/MN_FOK ./.

while unannotated text taken from the same corpus is as follows,

De keme1nyse1ge volt a legnagyobb .

References

Brill, E. 1992. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the DARPA Speech and Natural Language Workshop*. pp. 112-116. Morgan Kauffman. San Mateo, California.

Brill, E. 1993a. Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. pp. 259-265. ACL. Columbus, Ohio, USA.

Brill, E. 1993b. Transformation-Based Error-Driven Parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*. Tilburg, The Netherlands.

Brill, E. 1993c. *A Corpus Based Approach to Language Learning*. Ph.D. Dissertation, Department of Computer and Information Science, University of Pennsylvania.

Brill, E. 1994a. *A Report of Recent Progress in Transformation-Based Error-Driven Learning*. ARPA-94.

Brill, E. 1994b. Some Advances in Rule-Based Part of speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence* (AAAI-94), Seattle, Wa.

Brill, E. 1995a. Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. *Very Large Corpora Workshop*. 1995.

Brill, E. 1995b. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of speech Tagging. In *Computational Linguistics*. 21:4.

Brill, E. & Marcus, M. 1992a. Automatically Acquiring Phrase Structure Using Distributional Analysis. *DARPA Workshop on Speech and Natural Language*. 1992.

Brill, E. & Marcus, M. 1992b. Tagging an Unfamiliar Text With Minimal Human Supervision. In *Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language*. 1992.

Brill, E. & Resnik, P. 1994. A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation. In *Proceedings of Fifteenth International Conference on Computational Linguistics* COLING-1994. Kyoto, Japan.