Brill’s Tagger

Topic-wise study notes | Natural Language Processing

Transformation-based learning, initial tagging, and rule learning. Includes definitions, characteristics, architectures, algorithms, worked examples, and applications.

1. Definition and Overview

Brill’s Tagger is a part-of-speech (POS) tagging approach associated with Eric Brill. It begins with an initial tag assignment and applies an ordered sequence of learned transformation rules to correct errors.

POS tagging assigns a grammatical category to each token in context. For example, book is a noun in “a book” and a verb in “They book tickets”.

Main characteristics

• **Supervised learning:** standard training uses a manually tagged reference corpus.

• **Error-driven:** learns changes that improve the current tagging.

• **Rule-based output:** learned transformations are human-readable.

• **Context-sensitive:** rules can use nearby words and tags.

• **Greedy learning:** selects the best current rule at each iteration.

• **Ordered application:** later rules see the tagging produced by earlier rules.

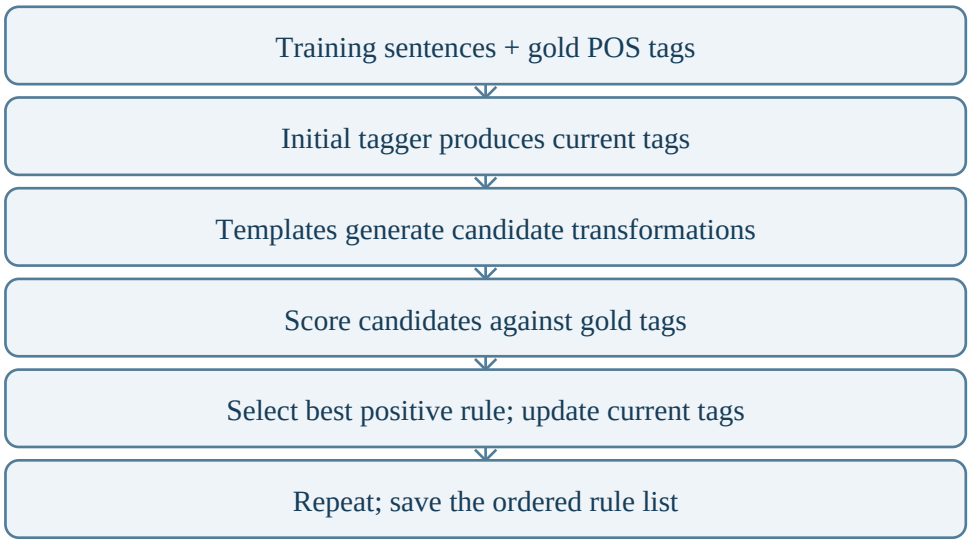
Common tags used in these notes

Tag	Meaning
NN	Singular or mass noun
NNS	Plural noun
VB	Base-form verb
VBP	Present-tense verb, other than third-person singular
VBZ	Present-tense verb, third-person singular
DT	Determiner
PRP	Personal pronoun
TO	The word to
JJ	Adjective

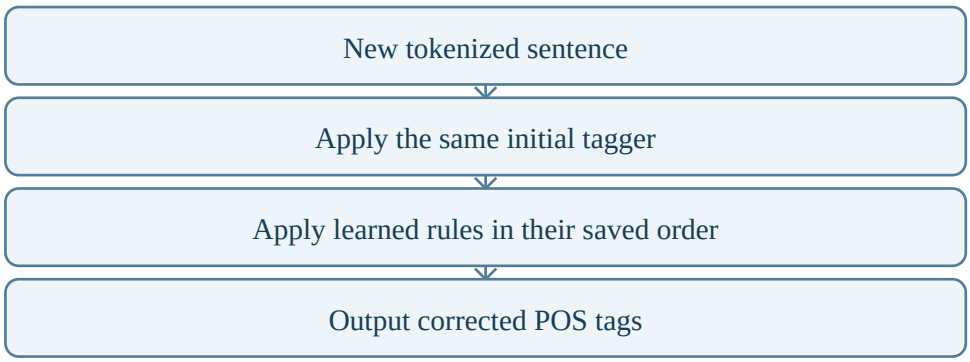
These labels follow common Penn Treebank conventions. Other tagsets use different labels. All numerical training examples below are illustrative, not reported experimental results.

2. Overall Architecture

Training architecture



Tagging new text



Why two stages are needed

The initial tagger makes a quick first prediction. Transformation rules then use context to repair systematic errors. Training needs gold tags to evaluate repairs; tagging new text does not require gold tags.

Main components

Component	Function
Tagged corpus	Supplies reference answers for training.
Initial tagger	Creates starting tags for training and new sentences.
Rule templates	Define the kinds of contextual conditions allowed.
Rule scorer	Counts errors corrected and correct tags damaged.
Rule learner	Selects and stores transformations greedily.
Rule executor	Applies saved transformations to current tags in order.

3. Transformation-Based Learning

Definition

Transformation-based learning (TBL) learns an ordered set of changes that improve an initial prediction. In POS tagging, each transformation changes one tag to another when a specified contextual condition holds.

Rule form: change tag A to tag B if condition C is true.
Example: NN → VB if the preceding tag is TO.

Characteristics

• Learns transformations from prediction errors rather than directly assigning every final tag.

• Uses predefined templates to restrict the search space.

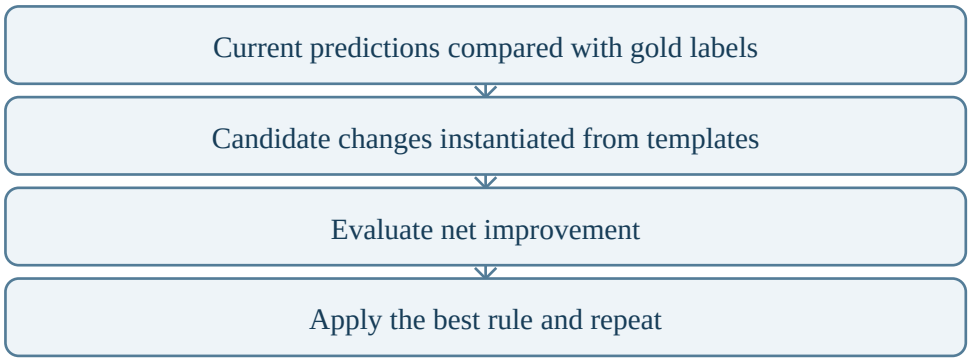
• Evaluates each rule over the current training tagging.

• Balances useful corrections against harmful changes.

• Updates the training state after selecting each rule.

• Can stop when no sufficiently beneficial rule remains.

Architecture



Rule score

Score(rule) = errors corrected - correct tags made incorrect. A change from one wrong tag to another wrong tag contributes zero to this simple token-accuracy score.

Effect	Contribution
Incorrect tag → correct tag	+1
Correct tag → incorrect tag	-1
Incorrect tag → another incorrect tag	0
No tag change	0

Applications

TBL can support POS tagging, text chunking, and other sequence-labeling tasks when initial labels, reference labels, and suitable transformation templates are available. POS tagging is the focus of Brill's Tagger here.

4. Initial Tagging

Definition

Initial tagging assigns the starting POS labels before any learned correction rules are applied. A common simple baseline gives a known word its most frequent tag in a training lexicon.

Characteristics

• Fast and relatively simple.

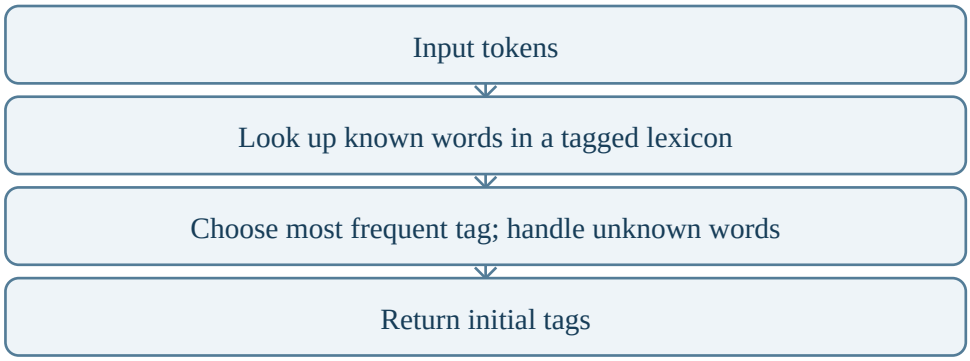
• Uses lexical frequency for known words in a basic implementation.

• Can mislabel ambiguous words because sentence context is limited.

• Needs an explicit policy for words absent from the lexicon.

• Must be consistent between training and inference.

Architecture



Step-by-step example

Sentence: “They want to book tickets.”
Assumed lexicon: book occurs most often as NN.

Token	Initial tag	Gold tag
They	PRP	PRP
want	VBP	VBP
to	TO	TO
book	NN	VB
tickets	NNS	NNS

• **Step 1:** Tokenize the sentence.

• **Step 2:** Retrieve the default tag for each known token.

• **Step 3:** Assign book/NN even though this context requires VB.

• **Step 4:** Pass the initial sequence to the transformation stage.

Unknown words and applications

A simple system may use a default NN tag for unknown words. Richer initializers can use capitalization, suffixes, or lexical rules. These are implementation choices.

5. Rule Learning

Definition

Rule learning discovers contextual transformations that improve the initial tagging on a gold-tagged training corpus. The output is an ordered list of rules, not a manually written correction list.

Characteristics

• Requires reference tags during standard supervised training.

• Uses templates to define permissible conditions.

• Generates rules that could repair observed errors.

• Selects the rule with the greatest current net gain.

• Applies that rule across the current corpus and rescans errors.

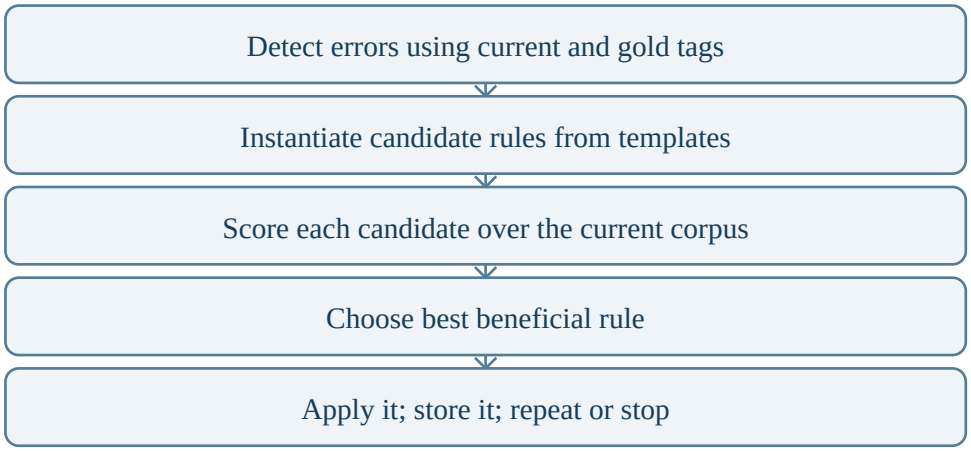
• Keeps rules in the order they were learned.

Template versus learned rule

Type	Example
Template	Change A to B when the previous tag is Z.
Instantiated rule	Change NN to VB when the previous tag is TO.
Template	Change A to B when the next tag is Z.
Instantiated rule	Change NN to JJ when the next tag is NN.

A template describes a family of possible rules. A learned rule fills in specific tags and conditions. Example rules are not universally valid: the training score determines whether a rule is useful on the current data.

Architecture



Applications

Rule learning captures reusable contextual patterns, supports explainable error analysis, and allows a tagger to adapt to a labelled domain corpus. Generalization still needs evaluation on held-out text.

6. Learning Algorithm and Scoring

Step-by-step training algorithm

- **1.** Build or choose the initial tagger using permitted training data.
- **2.** Initially tag the training sentences.
- **3.** Compare current labels with gold labels.
- **4.** Generate transformations allowed by templates.
- **5.** Score each candidate across all positions where it applies.
- **6.** Choose the highest-scoring candidate using a fixed tie policy.
- **7.** Stop if the rule does not meet the required positive-gain threshold.
- **8.** Apply the rule, append it to the rule list, and return to Step 3.

Worked scoring example

Suppose the current corpus has 10 tagging errors. Two candidate rules have the following effects:

Candidate	Fixed	Damaged	Score
R1: NN → VB after TO	4	1	3
R2: NN → VB after PRP	3	2	1

- **Step 1:** Compute R1 score: 4 - 1 = 3.
- **Step 2:** Compute R2 score: 3 - 2 = 1.
- **Step 3:** Select R1 because it has the higher net gain.
- **Step 4:** Apply R1 throughout the current training tagging.
- **Step 5:** Remaining errors = 10 - 4 + 1 = 7.
- **Step 6:** Recompute candidate effects; the old scores are no longer guaranteed to hold.

Why a rule may harm a correct tag

Contextual patterns are imperfect. In “I look forward to lunch”, lunch is NN and the preceding to can be tagged TO in Penn-style annotation. NN → VB after TO would damage lunch. Scoring must include such losses, not just the targeted corrections.

7. Worked Sentence: Error Correction

Input: “They want to book tickets.”
Learned rule used for illustration: NN → VB if the preceding tag is TO.

Step 1: Initial tagging

They	want	to	book	tickets
PRP	VBP	TO	NN	NNS

Book has the initial tag NN because its assumed most frequent lexical tag is noun.

Step 2: Check the rule conditions

Condition	Check
Current tag must be NN	book is currently NN: satisfied.
Previous tag must be TO	to is tagged TO: satisfied.
Replacement tag	VB, the base-form verb tag.

Step 3: Apply the transformation

Change book/NN to book/VB. Other tokens are unchanged by this rule.

Step 4: Updated tagging

They	want	to	book	tickets
PRP	VBP	TO	VB	NNS

Step 5: Continue the rule sequence

Apply any remaining learned rules in order to the updated tags. If none changes this sentence, the sequence above is the final output.

Interpretation

Book means reserve in this sentence, so it is a verb. In “They read a book”, book remains a noun because the illustrated rule’s preceding-tag condition is absent.

Training versus inference: During training, gold tags establish whether the change is correct. On new text, the system applies the saved rule without access to gold tags.

8. Rule Order and Pseudocode

Why ordering matters

Later rules operate on the labels produced by earlier transformations. A changed neighbouring tag can enable or disable another rule. The learned order must therefore be preserved.

Illustration: Start with “to/TO record/NN sales/NNS”. R1 changes NN → VB after TO, producing record/VB. A later rule conditioned on the previous tag being VB now sees VB before sales. This shows dependency between rules; it does not assert that sales needs a change.

Training pseudocode

```
TRAIN(tokens, gold_tags, initial_tagger, templates):
    current = initial_tagger(tokens)
    rules = []
    repeat:
        candidates = GENERATE(tokens, current,
                              gold_tags, templates)
        if candidates is empty: break
        best = rule with maximum NET_GAIN
        if NET_GAIN(best) < minimum_positive_gain:
            break
        current = APPLY(best, tokens, current)
        append best to rules
    return initial_tagger, rules
```

Tagging pseudocode

```
TAG(new_tokens, initial_tagger, learned_rules):
    tags = initial_tagger(new_tokens)
    for rule in learned_rules in saved order:
        tags = APPLY(rule, new_tokens, tags)
    return tags
```

A positive stopping threshold avoids zero-gain iterations. An implementation must define how a rule’s matching positions are collected and updated so training scores and execution use consistent semantics.

9. Applications and Limitations

Application	How POS tags help
Syntactic parsing	Provide grammatical categories for phrase and dependency analysis.
Information extraction	Help identify noun phrases and verbal relations.
Lemmatization	Help choose POS-appropriate lexical forms.
Corpus annotation	Supply automatic tags for inspection and correction.
Text chunking	Support grouping tokens into phrases; TBL can also learn chunk corrections.
Language teaching	Explain how context distinguishes noun and verb uses.
Domain adaptation	Learn corrections from a suitable tagged domain corpus.

Advantages

• Learned transformations are readable and traceable.

• Separates a simple baseline from contextual correction.

• Provides a clear framework for supervised error analysis.

• Can use word-based and tag-based contextual templates.

Limitations

• Training candidate search can be expensive.

• Quality depends on initial tags, gold annotation, and templates.

• Greedy selection does not guarantee the globally optimal rule sequence.

• Limited templates may miss long-range or semantic dependencies.

• Rare contexts and unknown words may remain difficult.

• Rules learned on one genre may not transfer well to another.

Evaluation

Evaluate on held-out text with a consistent tagset. Token accuracy equals correct predicted tags divided by evaluated tokens. Compare initial and final accuracy, inspect confusion patterns, and report unknown-word performance separately when useful. Do not learn the correction list from the test gold tags.

10. Revision and References

Short exam answer

Brill's Tagger is a transformation-based POS tagger. It first assigns initial tags, then learns contextual correction rules from a tagged training corpus. At each iteration, it selects the rule that maximizes errors corrected minus correct tags damaged, applies the rule, and repeats. New sentences are tagged using the initial tagger followed by the saved ordered rule list.

Practice questions

- 1. Define transformation-based learning.
- 2. Distinguish an initial tagger from a transformation rule.
- 3. What is the difference between a rule template and an instantiated rule?
- 4. A rule fixes 12 errors but damages 5 correct tags. Calculate its score.
- 5. Why must candidate scores be recalculated after applying a rule?
- 6. Trace book in "They want to book tickets".

Answer hints

1. Learning an ordered sequence of changes that improves initial predictions.
2. The initializer assigns starting tags; a transformation conditionally changes existing tags.
3. A template is a parameterized pattern; a rule fills in its specific values.
4. 12 - 5 = 7.
5. The current tags and rule conditions have changed.
6. Initial book/NN becomes book/VB under the illustrated rule NN → VB after TO.

Terminology checklist

Term	Meaning
Gold tags	Reference labels used for training or evaluation.
Current tags	The labels after the initializer and rules applied so far.
Transformation	A conditional change from one tag to another.
Net gain	Corrected errors minus damaged correct labels.
Ordered rule list	Learned transformations preserved in sequence.

Foundational reading

Brill, E. (1992). *A Simple Rule-Based Part of Speech Tagger*. Proceedings of the Third Conference on Applied Natural Language Processing, 152-155.
Brill, E. (1995). *Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging*. Computational Linguistics, 21(4), 543-565.

Architectures, counts, and sentence traces in these notes are illustrative teaching examples. Actual learned rules depend on the corpus, initializer, templates, and stopping settings.