Part-of-Speech Tagging

Rule-based, Stochastic and Transformation-based Tagging

Complete study notes: definition, key points, architecture, step-by-step examples and applications

Contents

1. Introduction to PoS Tagging
2. Rule-based Tagging
3. Stochastic Tagging
4. Transformation-based Tagging (Brill Tagger)
5. Comparison and Quick Revision

1. Introduction to PoS Tagging

Definition

Part-of-Speech (PoS) tagging is the process of assigning a grammatical category (noun, verb, adjective, adverb, determiner, preposition, etc.) to every word in a text, based on both the word itself and its context. The set of categories is called a **tagset**, for example the Penn Treebank tagset with 45 tags.

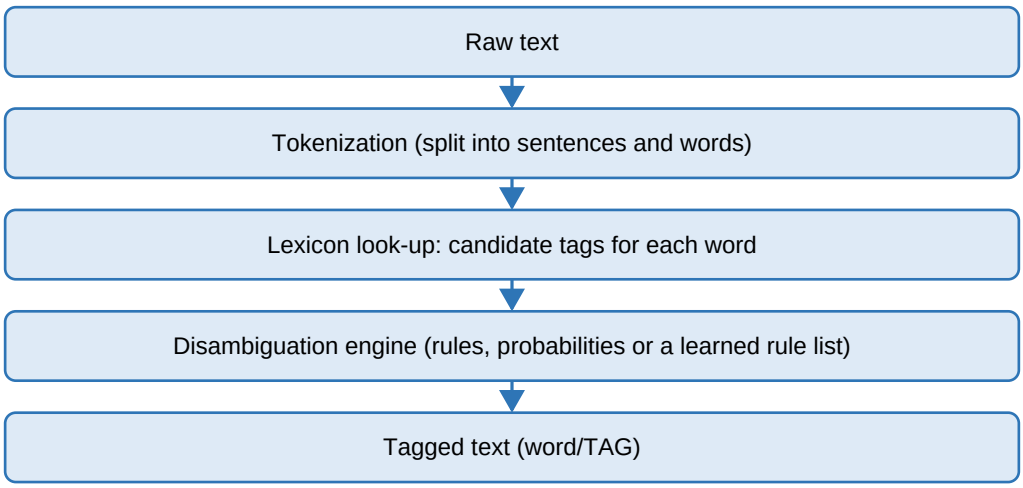
> **Input:** The dog barks loudly
> **Output:** The/DT dog/NN barks/VBZ loudly/RB

Important points

- **Ambiguity is the core problem.** Many words have more than one possible tag, and the tagger must pick the right one from the context.
- **Two sources of evidence:** the word itself (a lexical clue such as its most frequent tag or its suffix) and the context (the neighbouring words and tags).
- **Common tagsets:** Brown (87 tags), Penn Treebank (45 tags), Universal Dependencies (17 tags).
- **Baseline:** tagging every word with its most frequent tag already gives roughly 90% accuracy. Good taggers reach about 96 to 97%.
- **Unknown words** (not in the dictionary) are handled with suffix, prefix and capitalization clues.
- **Three classical families:** rule-based, stochastic (probabilistic) and transformation-based, which mixes both ideas.

Word	Possible tags	Example sentences
book	NN, VB	I read a **book** (NN). / **Book** that flight (VB).
can	MD, NN, VB	She **can** swim (MD). / A **can** of soda (NN).
race	NN, VB	The **race** was long (NN). / Expected to **race** tomorrow (VB).
light	NN, JJ, VB	Turn on the **light** (NN). / A **light** bag (JJ).

Architecture: general PoS tagger



Applications

- Syntactic parsing and chunking; named entity recognition and information extraction.
- Machine translation and word-sense disambiguation.
- Speech synthesis: *record* as noun and as verb are pronounced differently.
- Information retrieval, question answering, summarization and sentiment analysis.

2. Rule-based Tagging

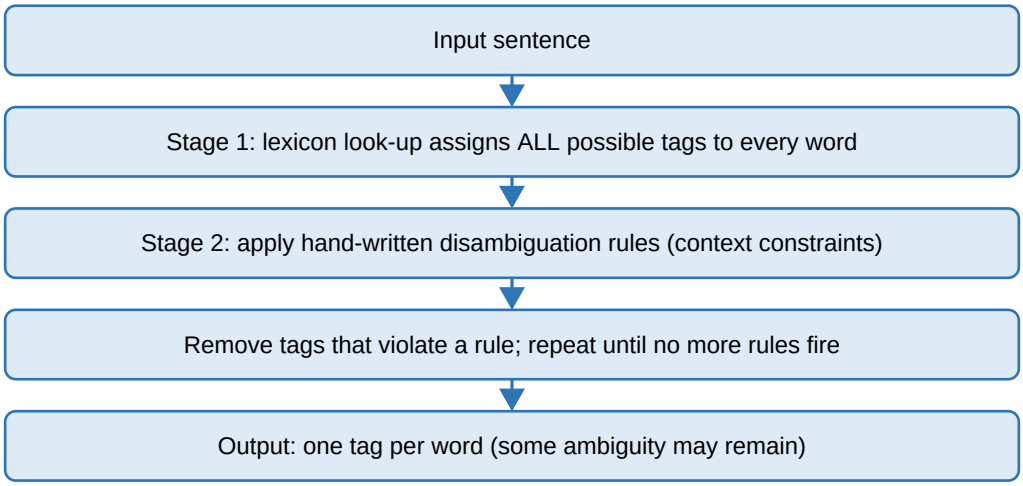
Definition

Rule-based tagging uses a dictionary (lexicon) to assign each word *all* its possible tags, and then applies a large set of **hand-written linguistic rules (constraints)** to eliminate incorrect tags until one tag remains per word. A well-known example is the **ENGTWOL** tagger, based on Constraint Grammar.

Important points

- Works in **two stages**: (1) dictionary look-up gives all candidates, (2) context rules remove the impossible ones.
- Rules are of the form *IF the context matches THEN eliminate (or choose) a tag*. Contexts look at neighbouring words and neighbouring tags.
- **Morphological rules** handle unknown words by suffix or capitalization (see table below).
- Knowledge comes from **linguists, not from a corpus**, so no training data is needed.
- Decisions are fully **explainable**: every removed tag can be traced to a rule.

Architecture



Component	Role
Lexicon and morphological analyzer	Stores words with their possible tags and analyses suffixes (-ing, -ed, -ly) for unknown words.
Rule base	Hand-crafted IF-THEN constraints written by linguists.
Rule engine	Applies rules in priority order and eliminates impossible tags.

Unknown-word clue	Rule	Example
Suffix -ness / -tion	Tag as noun (NN)	happiness, nation
Suffix -ly	Tag as adverb (RB)	quickly
Suffix -ing	Tag as verb gerund (VBG)	running
Capital letter inside sentence	Tag as proper noun (NNP)	Hyderabad

Example: step by step

Sentence: **“The can is empty.”**

Step 1: Tokenize: The | can | is | empty

Step 2: Dictionary look-up gives candidate tags:

Word	Candidate tags
The	DT
can	MD, NN, VB
is	VBZ
empty	JJ, VB

Step 3: Rule 1: IF the previous word is a determiner (DT), THEN eliminate verb readings (MD, VB). So “can” keeps only **NN**.

Step 4: Rule 2: IF the previous word is a form of “be” (VBZ), THEN eliminate the base-form verb (VB). So “empty” keeps only **JJ**.

Step 5: Output: The/DT can/NN is/VBZ empty/JJ

Advantages and limitations

- **Advantages:** transparent and easy to debug; high accuracy (about 95 to 99%) when the rules are good; no training corpus needed.
- **Limitations:** writing rules is slow and expensive; hard to port to a new language; rules may conflict; unseen constructions cause errors.

Applications

- Tagging morphologically rich languages (Finnish, Turkish, Hindi, Telugu) using linguistic knowledge.
- Grammar checkers and controlled-language checkers where explainable decisions are required.
- Bootstrapping annotation of new corpora (rule output is then corrected by hand).

3. Stochastic Tagging

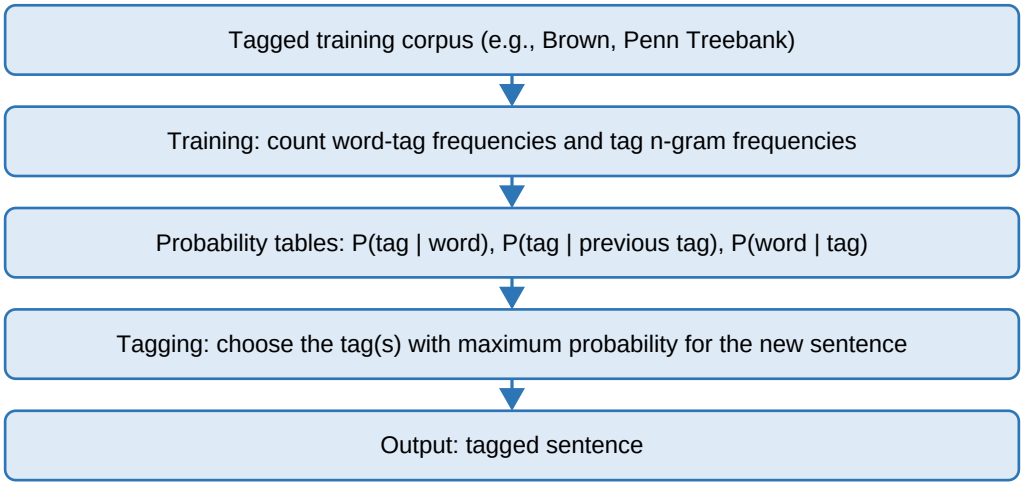
Definition

Stochastic (probabilistic) tagging uses a tagged training corpus to learn **statistics** about words and tags, and then chooses the most probable tag (or tag sequence) for new text. The simplest forms are the **word-frequency (unigram)** approach and the **tag-sequence (n-gram)** approach. Hidden Markov Model taggers are the standard advanced form.

Important points

- Probabilities are estimated by **counting** in a corpus (maximum likelihood estimation): P(t | w) = Count(w, t) / Count(w), and $P(t_i \mid t_{i-1}) = Count(t_{i-1}, t_i) / Count(t_{i-1})$.
- **Unigram** uses only the word; **bigram** adds the previous tag; **trigram** adds the two previous tags.
- The best tag sequence is found by maximizing P(t | previous tags) x P(word | t) over the sentence.
- **Data sparsity:** rare word-tag or tag-sequence pairs get zero counts, so **smoothing** (add-one, back-off, interpolation) is needed.
- Needs a large hand-tagged corpus (Brown, Penn Treebank) but little linguistic effort.

Architecture



Approach	Idea	Formula
Word-frequency (unigram)	Assign each word its most frequent tag in the corpus.	t* = argmax P(t | w)
Tag-sequence (bigram)	Use the previous tag as context together with the word.	$t^* = \text{argmax } P(t \mid t_{prev}) \times P(w \mid t)$
Trigram	Use the two previous tags as context.	$t^* = \text{argmax } P(t \mid t_{prev2}, t_{prev1}) \times P(w \mid t)$

Example 1: word-frequency tagging, step by step

Step 1: Collect counts for the word “race” in a tagged corpus: NN = 60, VB = 40 (total 100).

Step 2: Compute probabilities: P(NN | race) = 60/100 = 0.60; P(VB | race) = 40/100 = 0.40.

Step 3: Choose the maximum: the tagger always tags “race” as **NN**.

Step 4: **Problem:** in “expected to race tomorrow” the correct tag is VB. Context is needed, so we move to the bigram model.

Example 2: bigram tagging, step by step

Sentence: **“to race”**. The previous word “to” is tagged TO. Probabilities (illustrative values estimated from a corpus):

Quantity	Value
P(VB | TO)	0.83
P(NN | TO)	0.00047
P(race | VB)	0.00012
P(race | NN)	0.00057

Step 1: **Score for VB:** P(VB | TO) x P(race | VB) = 0.83 x 0.00012 = **0.000099**

Step 2: **Score for NN:** P(NN | TO) x P(race | NN) = 0.00047 x 0.00057 = **0.00000027**

Step 3: **Compare:** 0.000099 is much larger than 0.00000027, so the tagger selects **VB**: to/TO race/VB.

Advantages and limitations

- **Advantages:** learns automatically from data; gives probabilities; the most-frequent-tag baseline is already about 90% accurate.
- **Limitations:** needs a large tagged corpus; unseen words and sparse counts hurt accuracy; the context window is narrow; the model is hard to read.

Applications

- Quick baseline tagger and fast tagging in search engines and text-mining pipelines.
- Foundation for HMM taggers and for language models used in speech recognition.

4. Transformation-based Tagging (Brill Tagger)

Definition

Transformation-based tagging (Eric Brill, 1992 to 1995) combines the rule-based and stochastic ideas. It first assigns every word its most probable tag, then applies an **ordered list of transformation rules that were learned automatically** from a tagged corpus to correct the mistakes. Each rule has the form “change tag A to tag B when *condition*”. The method is also called **Transformation-Based Error-Driven Learning (TBL)**.

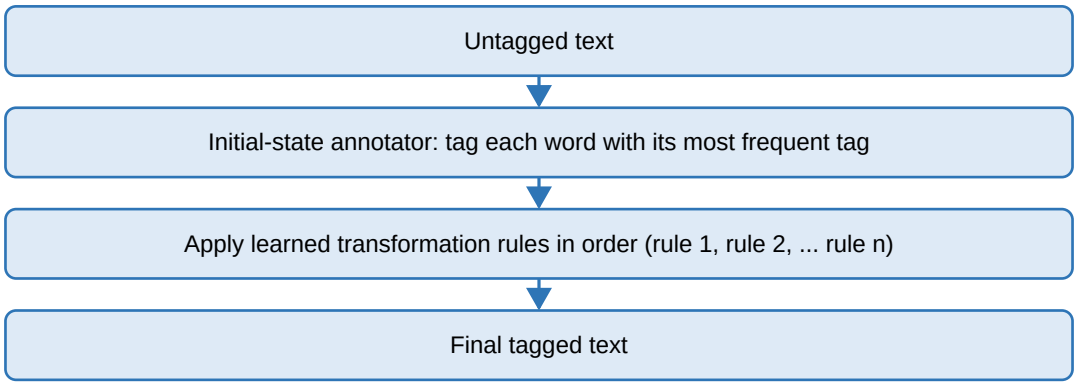
Important points

- **Initial-state annotator:** the starting tagging, usually the most frequent tag of each word.
- **Rule templates** restrict which rules can be generated (the previous tag, the next tag, one of the previous 3 tags, the previous word, etc.).
- **Error-driven learning:** in every iteration the rule that corrects the largest number of errors is selected and added to the list.

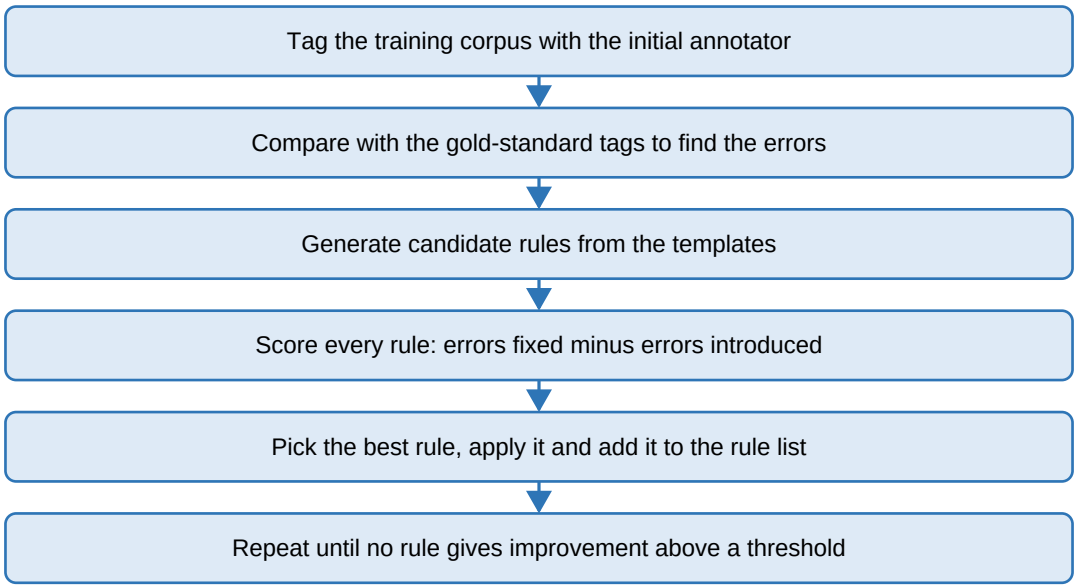
- **Order matters:** rules are applied one after another, and later rules can undo or refine earlier changes.
- The learned rules are **human-readable**, unlike probability tables. Accuracy is about 97%.

Architecture

Tagging phase:



Learning phase (training):



Template (examples)	Meaning
Previous tag is Z	Look at the tag of the word immediately before.
Next tag is Z	Look at the tag of the following word.
One of the previous 3 tags is Z	Wider left context window.
Previous word is W	Lexical condition.

Example 1: learning a rule (step by step)

Step 1: Initial tagging labels “race” as NN everywhere, since NN is its most frequent tag. Some of these are errors.

Step 2: **Candidate rules** are generated from the templates and each is scored on the corpus (illustrative figures):

Candidate rule	Errors fixed	Errors introduced	Net gain
NN to VB if previous tag is TO	120	5	115
NN to VB if previous tag is MD	90	3	87
NN to VB if next tag is DT	40	30	10

Step 3: **Select** the rule with the highest net gain: R1 = “NN to VB if previous tag is TO”, and apply it to the corpus.

Step 4: **Repeat:** rescore the remaining rules on the updated corpus. The next best becomes R2 = “NN to VB if previous tag is MD”. Stop when the gain is too small.

Example 2: tagging with the learned rules (step by step)

Learned rules in order: **R1:** NN to VB if previous tag is TO. **R2:** NN to VB if previous tag is MD.

Sentence 1: **“He wants to race tomorrow.”**

Step 1: **Initial tagging:** He/PRP wants/VBZ to/TO race/**NN** tomorrow/NN

Step 2: **Apply R1:** “race” is NN and the previous tag is TO, so change it to VB.

Step 3: **Apply R2:** no NN follows a modal, so there is no change.

Step 4: **Final:** He/PRP wants/VBZ to/TO race/**VB** tomorrow/NN

Sentence 2: **“She will book a flight.”**

Step 1: **Initial tagging:** She/PRP will/MD book/**NN** a/DT flight/NN

Step 2: **R1:** the previous tag of “book” is MD, not TO, so no change.

Step 3: **R2:** “book” is NN and the previous tag is MD, so change it to VB.

Step 4: **Final:** She/PRP will/MD book/**VB** a/DT flight/NN

Advantages and limitations

- **Advantages:** rules are learned automatically yet stay readable; accuracy about 97%; small model and fast tagging.
- **Limitations:** training is slow because many candidate rules are tested; rules are applied in strict order, so no probability scores are produced.

Applications

- General English tagging (the Brill tagger is available in NLTK).
- Domain adaptation: learning rules for biomedical or legal text.
- Other tasks solved with TBL: parsing, chunking, word-sense disambiguation and spelling correction.

5. Comparison and Quick Revision

Aspect	Rule-based	Stochastic	Transformation-based
Knowledge source	Hand-written rules	Corpus statistics (probabilities)	Rules learned from a corpus
Needs tagged corpus	No	Yes	Yes
Starting point	All possible tags per word	Counts and probability tables	Most frequent tag per word
How it decides	Eliminates tags using rules	Picks the highest probability	Applies ordered correction rules
Output	One tag after elimination	Most probable tag or sequence	Tag after the rule sequence
Interpretability	Very high	Medium	High
Typical accuracy	95 to 99% (well engineered)	About 90 to 95%	About 97%
Main drawback	Costly to build by hand	Needs large data, narrow context	Slow training

Quick revision points

- PoS tagging assigns a grammatical category to every word and must resolve ambiguity from the context.
- **Rule-based:** dictionary gives all tags, hand-written rules eliminate wrong ones (ENGTWOL).
- **Stochastic:** counts from a tagged corpus; unigram uses P(t | w), bigram uses P(t | previous tag) x P(w | t).
- **Transformation-based:** most frequent tag first, then learned correction rules applied in order (Brill).
- Stochastic models extend naturally to HMM taggers (Viterbi decoding), and TBL combines the best of rules and statistics.

Summary: Rule-based tagging offers precision through linguistic knowledge, stochastic tagging adds data-driven probability, and transformation-based tagging learns readable rules from data, reaching about 97% accuracy.