NLP UNIT-2: English Word Classes, POS Tags, Types of POS Tagging

English Word Classes

Traditionally the definition of parts-of-speech has been based on syntactic and morphological function; words that function similarly with respect to what can occur nearby (their "syntactic distributional properties"), or with respect to the affixes they take (their morphological properties) are grouped into classes. While word classes do have tendencies toward semantic coherence (nouns do in fact often describe "people, places or things", and adjectives often describe properties), this is not necessarily the case, and in general we don't use semantic coherence as a definitional criterion for parts-of-speech.

Parts-of-speech can be divided into two broad supercategories: **closed class** types and **open class** types. Closed classes are those that have relatively fixed membership. For example, prepositions are a closed class because there is a fixed set of them in English; new prepositions are rarely coined. By contrast nouns and verbs are open classes because new nouns and verbs are continually coined or borrowed from other languages (e.g., the new verb *to fax* or the borrowed noun *futon*). It is likely that any given speaker or corpus will have different open class words, but all speakers of a language, and corpora that are large enough, will likely share the set of closed class words. Closed class words are also generally **function words** like *of*, *it*, *and*, or *you*, which tend to be very short, occur frequently, and often have structuring uses in grammar.

There are four major open classes that occur in the languages of the world; **nouns**, **verbs**, **adjectives**, and **adverbs**. It turns out that English has all four of these, although not every language does.

NOUN **Noun** is the name given to the syntactic class in which the words for most people, places, or things occur. But since syntactic classes like **noun** are defined syntactically and morphologically rather than semantically, some words for people, places, and things may not be nouns, and conversely some nounsmay not be words for people, places, or things. Thus nouns include concrete terms like *ship* and *chair*, abstractions like *bandwidth* and *relationship*, and verb-like terms like *pacing* as in *His pacing to and fro became quite annoying*. What defines a noun in English, then, are things like its ability to occur with determiners (*a goat, its bandwidth, Plato's Republic*), to take possessives (*IBM's annual revenue*), and for most but not all nouns, to occur in the plural form (*goats, abaci*).

Nouns are traditionally grouped into **proper nouns** and **common nouns**. Proper nouns, like *Regina*, *Colorado*, and *IBM*, are names of specific persons or entities. In English, they generally aren't preceded by articles (e.g., *the book is upstairs*, but *Regina is upstairs*). In written English, proper nouns are usually capitalized.

In many languages, including English, common nouns are divided into **count nouns** and **mass nouns**. Count nouns are those that allow grammatical enumeration; that is, they can occur in both the singular and plural (*goat/goats, relationship/relationships*) and they can be counted (*one goat, two goats*). Mass nouns are used when something is conceptualized as a homogeneous group. So words like *snow, salt*, and *communism* are not counted (i.e., **two snows* or **two communisms*). Mass nouns can also appear without articles where singular count nouns cannot (*Snow is white* but not **Goat is white).*

The **verb** class includes most of the words referring to actions and processes, including main verbs like draw, provide, differ, and go. English verbs have a number of morphological forms (non-3rd-person-sg (eat), 3rd-person-sg (eats), progressive (eating), past participle (eaten)). A subclass of English verbs called **auxiliaries** will be discussed when we turn to closed class forms. While many researchers believe that all human languages have the categories of noun and verb, others have argued that some languages, such as Riau Indonesian and Tongan, don't even make this distinction (Broschart, 1997; Evans, 2000; Gil, 2000).

The third open class English form is adjectives; semantically this class includes many terms that describe properties or qualities. Most languages have adjectives for the concepts of color (white, black), age (old, young), and value (good, bad), but there are languages without adjectives. In Korean, for example, the words corresponding to English adjectives act as a subclass of verbs, so what is in English an adjective 'beautiful' acts in Korean like a verb meaning 'to be beautiful' (Evans, 2000).

The final open class form, **adverbs**, is rather a hodge-podge, both semantically and formally. For example Schachter (1985) points out that in a sentence like the following, all the italicized words are adverbs:

Unfortunately, John walked home extremely slowly yesterday

What coherence the class has semantically may be solely that each of these words can be viewed as modifying something (often verbs, hence the name "adverb", but also other adverbs and entire verb phrases). **Directional adverbs** or **locative adverbs** (home, here, downhill) specify the direction or location of some action; **degree adverbs** (extremely, very, somewhat) specify the extent of some action, process, or property; **manner adverbs** (slowly, slinkily, delicately) describe the manner of some action or process; and **temporal adverb** describe the time that some action or event took place (yesterday, Monday). Because of the heterogeneous nature of this class, some adverbs (for example temporal adverbs like Monday) are tagged in some tagging schemes asnouns.

The closed classes differ more from language to language than do the open classes.

Here's a quick overview of some of the more important closed classes in English, with a few examples of each:

• **prepositions:** on, under, over, near, by, at, from, to, with

• **determiners:** a, an, the

• **pronouns:** she, who, I, others

• **conjunctions:** and, but, or, as, if, when

• **auxiliary verbs:** can, may, should, are

• **particles:** up, down, on, off, in, out, at, by,

• **numerals:** one, two, three, first, second, third

- **Prepositions** occur before noun phrases; semantically they are relational, often indicating spatial or temporal relations, whether literal (on it, before then, by the house) or metaphorical (on time, with gusto, beside herself). But they often indicate other relations as well (Hamlet was written by Shakespeare, and [from Shakespeare] "And I did laugh sans intermission an hour by his dial").

A **particle** is a word that resembles a preposition or an adverb, and is used icombination with a verb. When a verb and a particle behave as a single syntactic and/or semantic unit, we call the combination a **phrasal verb**. Phrasal verbs can behave as a semantic unit; thus they often have a meaning that is not predictable from the separate meanings of the verb and the particle. Thus *turn down* means something like 'reject', *rule out* means 'eliminate', *find out* is 'discover', and *go on* is 'continue'; these are no meanings that could have been predicted from the meanings of the verb and the particle independently. Here are some examples of phrasal verbs from Thoreau:

So I *went on* for some days cutting and hewing timber. . .

Moral reform is the effort to *throw off* sleep. . .

Particles don't always occur with idiomatic phrasal verb semantics; here are more examples of particles from the Brown corpus:

> . . . she had turned the paper *over*.
> He arose slowly and brushed himself *off*.
> He packed *up* his clothes.

A closed class that occurs with nouns, often marking the beginning of a noun phrase, is the **determiners**. One small subtype of determiners is the **articles**: English has three articles: *a*, *an*, and *the*. Other determiners include *this* (as in *this chapter*) and *that* (as in *that page*).

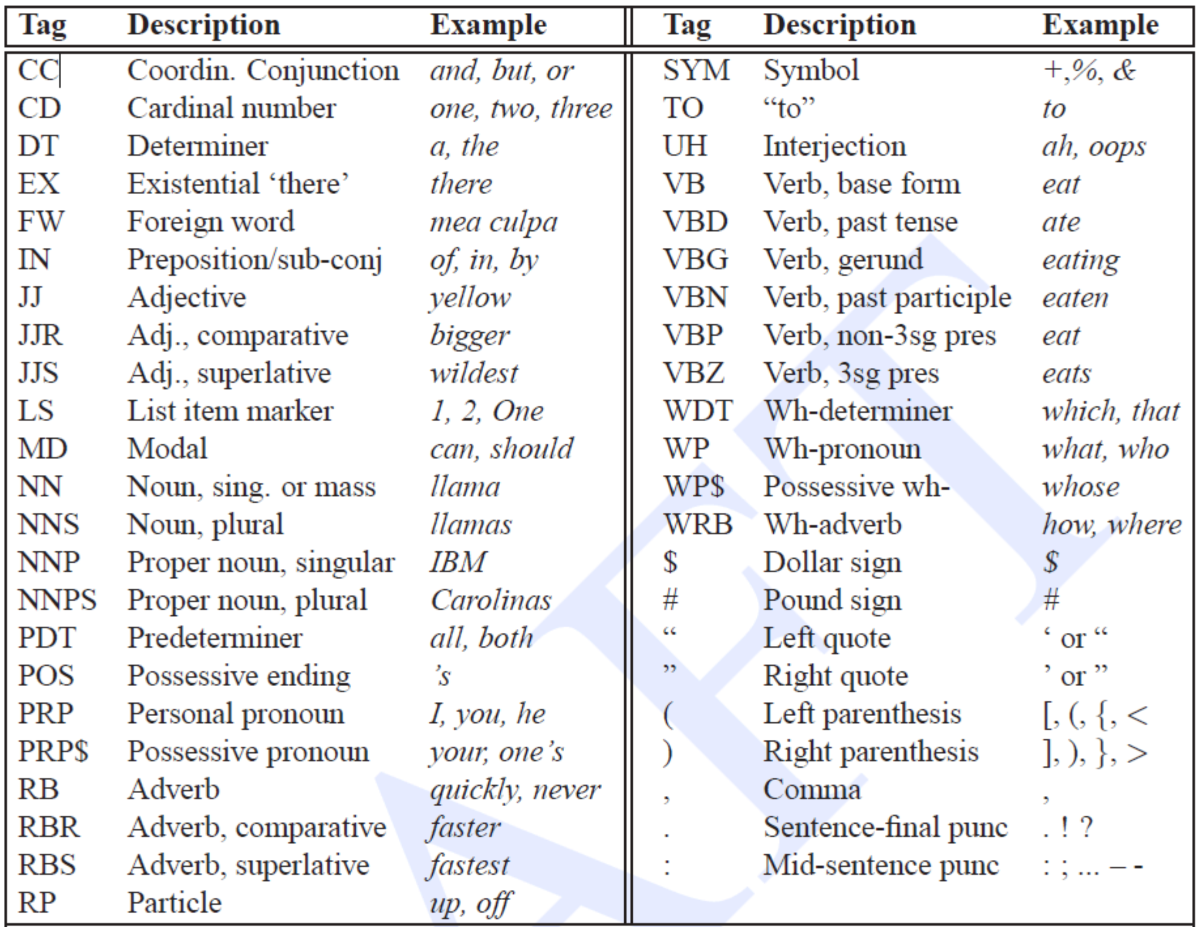
Conjunctions are used to join two phrases, clauses, or sentences. Coordinating conjunctions like *and*, *or*, and *but*, join two elements of equal status. Subordinating conjunctions are used when one of the elements is of some sort of embedded status. For example *that* in *"I thought that you might like some milk"* is a subordinating conjunction that links the main clause *I thought* with the subordinate clause *you might like some milk*. This clause is called subordinate because this entire clause is the "content" of the main verb *thought*. Subordinating conjunctions like *that* which link a verb to its argument in this way are also called **complementizers.**

Pronouns are forms that often act as a kind of shorthand for referring to some noun phrase or entity or event. **Personal pronouns** refer to persons or entities (*you*, *she*, *I*, *it*, *me*, etc.).

Possessive pronouns are forms of personal pronouns that indicate either actual possession or more often just an abstract relation between the person and some object (*my, your, his, her, its, one's, our, their*). **Wh-**WH **pronouns** (*what, who, whom, whoever*) are used in certain question forms, or may also act as complementizers (*Frieda, who I met five years ago* . . .).

A closed class sub type of English verbs are the **auxiliary** verbs. Cross linguistically, auxiliaries are words (usually verbs) that mark certain semantic features of a main verb, including whether an action takes place in the present, past or future (tense), whether it is completed (aspect), whether it is negated (polarity), and whether an action is necessary, possible, suggested, desired, etc. (mood). English auxiliaries include the **copula** verb *be*, the two verbs *do* and *have*, along with their inflected forms, as well as a class of **modal verbs**. *Be* is called a copula because it connects subjectswith certain kinds of predicate nominals and adjectives (*He is a duck*). The verb *have* is used for example to mark the perfect tenses (*I have gone*, *I had gone*), while *be* is used as part of the passive (*We were robbed*), or progressive (*We are leaving*) constructions. The modals are used to mark the mood associated with the event or action depicted by the main verb. So *can* indicates ability or possibility, *may* indicates permission or possibility, *must* indicates necessity, and so on.

English also has many words of more or less unique function, including **interjections** (*oh, ah, hey, man, alas, uh, um*), **negatives** (*no, not*), **politeness markers** (*please, thank you*), **greetings** (*hello, goodbye*), and the existential **there** (*there are two on the table*) among others. Whether these classes are assigned particular names or lumped together (as interjections or even adverbs) depends on the purpose of the labeling.



Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	‘ or “
POS	Possessive ending	*’s*	”	Right quote	’ or ”
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	[, (, {, <
PRP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	],), }, >
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	. ! ?
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	: ; ... – -
RP	Particle	*up, off*			

Examples

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

There/EX are/VBP 70/CD children/NNS **there/RB**

Although/IN preliminary/JJ findings/NNS were/VBD **reported/VBN** more/RBR than/IN a/DT year/NN ago/IN ,/, the/DT latest/JJS results/NNS appear/VBP in/IN today/NN **’s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ,/,

Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG

All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN

Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD

income-tax/JJ return/NN

the/DT Gramm-Rudman/NP Act/NP

Pacific/NN waters/NNS

POS Tagging

POS Tagging (or just tagging for short) is the process TAGGING of assigning a partof-speech or other syntactic class marker to each word in a corpus. Because tags are generally also applied to punctuation, tagging requires that the punctuation marks (period, comma, etc) be separated off of the words. Thus tokenization is usually performed before, or as part of, the tagging process, separating commas, quotation marks, etc., from words, and disambiguating end-of-sentence punctuation (period, question mark, etc) from part-of-word punctuation (such as in abbreviations like *e.g.* and *etc.*). The input to a tagging algorithm is a string of words and a specified tagset of the kind described in the previous section. The output is a single best tag for each word.

Rule-Based POS Tagging

The earliest algorithms for for automatically assigning part-of-speechwere based on a twostage architecture (Harris, 1962; Klein and Simmons, 1963; Greene and Rubin, 1971). The first stage used a dictionary to assign each word a list of potential parts-of-speech. The second stage used large lists of hand-written disambiguation rules to winnow down this list to a single part-of-speech for each word.

Modern rule-based approaches to part-of-speech tagging have a similar architecture, although the dictionaries and the rule sets are vastly larger than in the 1960's.

Components of a Rule-Based Tagger

1. **Lexicon / Dictionary**

o A list of words and their possible POS tags.

o Example:

▪ "book" → noun, verb

▪ "run" → noun, verb

2. **Rules**

o Linguistic rules applied to resolve ambiguity.

o Types of rules:

▪ **Contextual Rules**: Use surrounding words to decide the tag.

▪ Example: If a word follows a determiner (DT), tag it as a noun (NN).

▪ **Morphological Rules**: Use word suffix/prefix patterns.

▪ Example: Words ending in -ing → verb (VBG).

▪ **Fallback Rules**: Default to the most common tag in lexicon if no other rules apply.

Working of a Rule-Based Tagger

1. **Look up each word** in the dictionary for possible POS tags.
2. **Apply disambiguation rules** based on context or morphology.
3. **Assign the most appropriate tag** to each word.

Example:

Sentence: "The cat sleeps on the mat."

• Lexicon lookup:

o "The" → DT

o "cat" → NN

o "sleeps" → VB, NNS

• Rule application:

o If previous word = DT, current word = NN → "cat" tagged as NN

o "sleeps" follows NN → likely VB → tagged as VB

Output: "The/DT cat/NN sleeps/VB on/IN the/DT mat/NN ./"

4. Advantages

• **No training data required**

• Can be **very accurate** for well-defined domains

• Easy to **interpret and debug**

5. Disadvantages

• **Labor-intensive**: Rules must be manually crafted for each language.

• **Limited coverage**: Cannot handle all lexical ambiguities or unknown words.

• **Not scalable** for large corpora or multiple languages.

• **Context limitation**: Cannot capture long-range dependencies like neural models.

6. Applications

• Early **POS tagging systems** in English and other languages.

• Useful in **domain-specific NLP systems** where training data is scarce.

Stochastic (Probabilistic) POS Tagging

Stochastic POS Tagging assigns part-of-speech tags based on **probabilities derived from annotated corpora**.

- It is also called **statistical POS tagging**.
- Uses **contextual information** to resolve ambiguities that rule-based methods may fail to handle.

Use of a Hidden Markov Model to do part-of-speech-tagging, as we will define it, is a special case of **Bayesian inference**, a paradigmthat has been known since the work of Bayes (1763). Bayesian inference or Bayesian classification was applied successfully to language problems as early as the late 1950s, including the OCR work of Bledsoe in 1959, and the seminal work of Mosteller and Wallace (1964) on applying Bayesian inference to determine the authorship of the Federalist papers.

In a classification task, we are given some observation(s) and our job is to determine which of a set of classes it belongs to. Part-of-speech tagging is generally treated as a sequence classification task. So here the observation is a sequence of words (let's say a sentence), and it is our job to assign them a sequence of part-of-speech tags.

For example, say we are given a sentence like

Secretariat is expected to race tomorrow

The Bayesian interpretation of this task starts by considering all possible sequences of classes—in this case, all possible sequences of tags. Out of this universe of tag sequences, we want to choose the tag sequence which is most probable given the observation sequence of *n* words *wn*1 . In other words, we want, out of all sequences of *n* tags *tn* 1 the single tag sequence such that *P*(*tn*1 |*wn*1ˆ) is highest. We use the hat notation ˆ to mean "our estimate of the correct tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

The function argmax(X) means "the *x* such that *f* (*x*) is maximized". Equation thus means, out of all tag sequences of length *n*, we want the particular tagsequence *tn* 1 which maximizes the right-hand side. While it is guaranteed to give us the optimal tag sequence, it is not clear how to make the equation operational; that is, for a given tag sequence *tn* 1 and word sequence *wn*1 , we don't know how to directly compute *P*(*tn*1 |*wn*1).

The intuition of Bayesian classification is to use Bayes' rule to transform into a set of other probabilities which turn out to be easier to compute. Bayes' rule is presented in the below formula; it gives us a way to break down any conditional probability *P*(*x*|*y*) into three other probabilities:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

We can then substitute in the above formula to get

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

Since we are choosing a tag sequence out of all tag sequences, we will be computing *P*(*wn*1 |*tn* 1)*P*(*tn*1) / *P*(*wn*1) for each tag sequence. But *P*(*wn*1) doesn't change for each tag sequence; we are always asking about the most likely tag sequence for the same observation *wn*1 ,which must have the same probability *P*(*wn*1). Thus we can choose the tag sequence which maximizes this simpler formula

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

To summarize, the most probable tag sequence ˆ*tn* 1 given some word string *wn*1 can be computed by taking the product of two probabilities for each tag sequence, and choosing the tag sequence for which this product is greatest. The two terms are the **prior probability** of the tag sequence *P*(*tn*1)), and the **likelihood** of the word string LIKELIHOOD *P*(*wn*1|*tn*1)

Unfortunately, the above formula is still too hard to compute directly. HMM taggers therefore make two simplifying assumptions. The first assumption is that the probability of a word appearing is dependent only on its own part-of-speech tag; that it is independent of other words around it, and of the other tags around it:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

The second assumption is that the probability of a tag appearing is dependent only on the previous tag, the **bigram** assumption

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Plugging the simplifying assumptions into above argmax formula results in thefollowing equation by which a bigramtagger estimates themost probable tag sequence:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Example:

Sentence

Secretariat is expected to race tomorrow

Tokens

w1=Secretariat, w2=is, w3=expected, w4=to, w5=race, w6=tomorrow

HMM Components

In **HMM POS tagging**:

- **States (T)** → POS tags
- **Observations (W)** → words
- **Transition Probability**

$$P(t_i|t_{i-1})$$

- **Emission Probability**

$$P(w_i|t_i)$$

The **Viterbi algorithm** finds:

$$\hat{t}_{1:n} = argmax\prod P(w_i|t_i)P(t_i|t_{i-1})$$

Candidate POS Tags (restricted for clarity)

Word	**Possible Tags**
Secretariat	NNP
Is	VBZ
expected	VBN
To	TO
Race	VB, NN
tomorrow	NN, RB

Assumed Probabilities (Typical Corpus-Based Values)

Emission Probabilities

Word	**Tag**	**P(word | tag)**
Secretariat	NNP	0.90
Is	VBZ	0.95
expected	VBN	0.60
To	TO	0.99
Race	VB	0.50
Race	NN	0.30
tomorrow	NN	0.40
tomorrow	RB	0.45

Transition Probabilities

From → To	**Probability**
START → NNP	0.40
NNP → VBZ	0.50
VBZ → VBN	0.40
VBN → TO	0.60

From → To	Probability
TO → VB	0.80
TO → NN	0.05
VB → NN	0.40
VB → RB	0.30

Viterbi Table Calculations

Step 1: Initialization (Word 1)

$$V_1(NNP) = P(NNP|START) \times P(Secretariat|NNP)$$

Tag	Calculation	Score
NNP	0.40 × 0.90	**0.36**

Step 2: Word 2 = *is*

$$V_2(VBZ) = V_1(NNP) \times P(VBZ|NNP) \times P(is|VBZ)$$

Tag	Calculation	Score
VBZ	0.36 × 0.50 × 0.95	**0.171**

Step 3: Word 3 = *expected*

Tag	Calculation	Score
VBN	0.171 × 0.40 × 0.60	**0.04104**

Step 4: Word 4 = *to*

Tag	Calculation	Score
TO	0.04104 × 0.60 × 0.99	**0.02437**

Step 5: Word 5 = *race*

Two candidate tags:

VB

$$0.02437 \times 0.80 \times 0.50 = 0.00975$$

NN

$$0.02437 \times 0.05 \times 0.30 = 0.00037$$

Choose VB (higher probability)

Step 6: Word 6 = *tomorrow*

Two candidate tags:

NN

$$0.00975 \times 0.40 \times 0.40 = 0.00156$$

RB

$$0.00975 \times 0.30 \times 0.45 = 0.00132$$

Choose NN

Final Best Tag Sequence (Backtracking)

Secretariat/NNP

is/VBZ

expected/VBN

to/TO

race/VB

tomorrow/NN

Transformation Based Tagging

Transformation-Based Tagging, sometimes called Brill tagging, is an instance of the **Transformation-Based Learning** (TBL) approach to machine learning (Brill, 1995), and draws inspiration from both the rule-based and stochastic taggers. Like the rulebased taggers, TBL is based on rules that specify what tags should be assigned to what words. But like the stochastic taggers, TBL is a machine learning technique, in which rules are automatically induced from the data. Like some but not all of the HMMtaggers, TBL is a supervised learning technique; it assumes a pre-tagged training corpus.

Samuel et al. (1998) offer a useful analogy for understanding the TBL paradigm, which they credit to Terry Harvey. Imagine an artist painting a picture of a white house with green trim against a blue sky. Suppose most of the picture was sky, and hence most of the picture was blue. The artist might begin by using a very broad brush and painting the entire canvas blue. Next she might switch to a somewhat smaller white brush, and paint the entire house white. She would just color in the whole house, not worrying about the brown roof, or the blue windows or the green gables. Next she takes a smaller brown brush and colors over the roof. Now she takes up the blue paint on a small brush and paints in the blue windows on the house. Finally she takes a very fine green brush and does the trim on the gables.

The painter starts with a broad brush that covers a lot of the canvas but colors a lot of areas that will have to be repainted. The next layer colors less of the canvas, but also makes less "mistakes". Each new layer uses a finer brush that corrects less of the picture, but makes fewer mistakes. TBL uses somewhat the same method as this painter. The TBL algorithm has a set of tagging rules. A corpus is first tagged using the broadest rule, that is, the one that applies to the most cases. Then a slightly more specific rule is chosen, which changes some of the original tags. Next an even narrower rule, which changes a smaller number of tags (some of which might be previously changed tags).

How TBL Rules are Applied

Let's look at one of the rules used by Brill's (1995) tagger. Before the rules apply, the tagger labels every word with its most-likely tag. We get these most-likely tags from a tagged corpus. For example, in the Brown corpus, *race* is most likely to be a noun:

$P(\text{NN}|\text{race}) = .98$

$P(\text{VB}|\text{race}) = .02$

This means that the two examples of *race* that we saw above will both be coded as NN. In the first case, this is a mistake, as NN is the incorrect tag:

Secretariat/NN VBZ expected/VBN to/TO race/**NN** tomorrow/NN

In the second case this *race* is correctly tagged as an NN:

the/DT race/**NN** for/IN outer/JJ space/NN

After selecting the most-likely tag, Brill's tagger applies its transformation rules. As it happens, Brill's tagger learned a rule that applies exactly to this mistagging of *race*:

Change NN *to* VB *when the previous tag is* TO This rule would change *race/NN* to *race/VB* in exactly the following situation, since it is preceded by *to/TO*:

expected/VBN to/TO race/NN→ expected/VBN to/TO race/VB

Maximum Entropy Model for POS Tagging

MaxEnt models compute **probabilities of tags using rich features** from words and context.

• Tag with **maximum probability** is assigned.

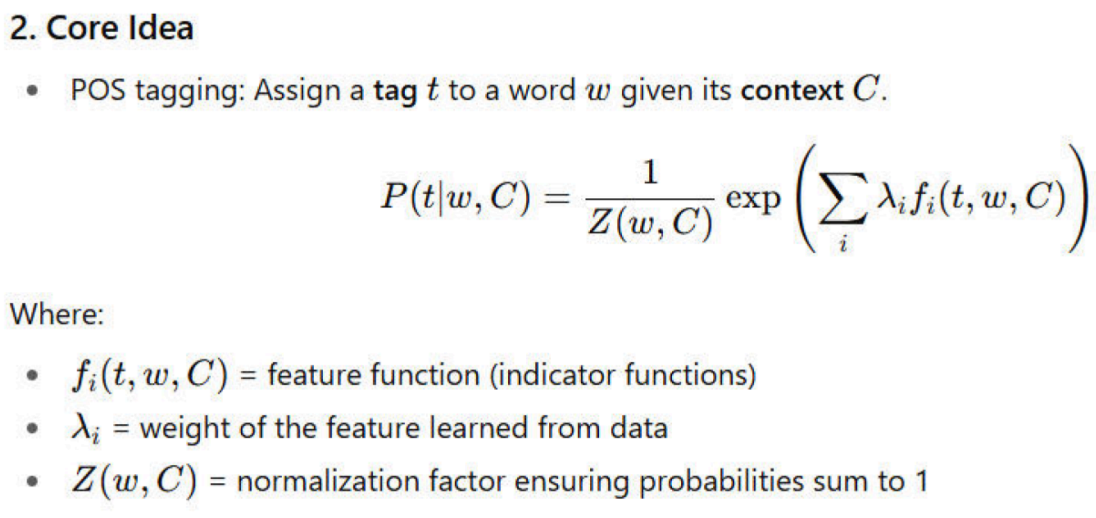
• Advantages over HMM: **flexible, feature-rich, no strong independence assumptions**.

1. Introduction

• Maximum Entropy Models are probabilistic models used in NLP for sequence labeling tasks like POS tagging.

• Based on the principle of maximum entropy: among all probability distributions satisfying given constraints, choose the one with highest entropy (most uniform / least biased).

Advantages: Can incorporate diverse features, not limited to sequential dependencies like HMMs.



2. Core Idea

- POS tagging: Assign a **tag** t to a word w given its **context** C.

$$P(t|w,C) = \frac{1}{Z(w,C)} \exp\left(\sum_i \lambda_i f_i(t,w,C)\right)$$

Where:

- $f_i(t,w,C)$ = feature function (indicator functions)
- λ_i = weight of the feature learned from data
- $Z(w,C)$ = normalization factor ensuring probabilities sum to 1

3. Features Used in POS Tagging

MaxEnt models allow **rich features**, such as:

1. **Lexical Features**

o Current word, suffixes, prefixes, capitalization

o Example: If word ends in -ing, likely VB

2. **Contextual Features**

o Previous and next words or tags

o Example: If previous word = to, current word → VB

3. **Orthographic Features**

o Numbers, hyphens, punctuation

o Example: If word contains digits → NN (numeric)

4. **Combined Features**

o Previous tag + current word, word shape, etc.

4. How MaxEnt POS Tagging Works

1. **Training Phase**

o Input: Annotated corpus (words + correct tags)

o Learn weights (λi\lambda_iλi) for each feature to maximize likelihood

2. **Tagging Phase**

o For each word:

▪ Extract features from word and context

▪ Compute probabilities of all possible tags

▪ Assign tag with **highest probability**

5. Example

Sentence: "The cat sleeps"

Features for "sleeps":

- Current word = "sleeps"
- Previous word = "cat"
- Previous tag = "NN"
- Word suffix = "ps"

Compute probability for each candidate tag:

- P(VB | features) = 0.75
- P(NN | features) = 0.10
- P(JJ | features) = 0.05

→ Assign **VB** as tag for "sleeps" because it has **highest probability**.

6. Advantages

- Can incorporate **arbitrary, overlapping features**.
- Does **not require independence assumptions** like HMMs.
- Often achieves **higher accuracy** in POS tagging than HMMs.

7. Disadvantages

- Computationally **more expensive** than HMMs for large feature sets.
- Requires **good feature engineering** (though deep learning reduces this need).
- Needs a **large annotated corpus** for robust performance.

8. Applications

- **POS Tagging** (main application)
- **Named Entity Recognition (NER)**
- **Chunking / Shallow Parsing**
- **Information Extraction**

Issues with POS Tagging

Tag Indeterminacy and Tokenization

Tag indeterminacy arises when a word is ambiguous between multiple tags and it is impossible or very difficult to disambiguate. In this case, some taggers allow the use of multiple tags. This is the case in both the Penn Treebank and in the British National Corpus. Common tag indeterminacies include adjective versus preterite versus past participle (JJ/VBD/VBN), and adjective versus noun as prenominal modifier (JJ/NN). Given a corpus with these indeterminate tags, there are 3 ways to deal with tag indeterminacy when training and scoring part-of-speech taggers:

1. Somehow replace the indeterminate tags with only one tag.

2. In testing, count a tagger as having correctly tagged an indeterminate token if it gives either of the correct tags. In training, somehow choose only one of the tags for the word.

3. Treat the indeterminate tag as a single complex tag.

The second approach is perhaps the most sensible, although most previous published results seem to have used the third approach. This third approach applied to the Penn Treebank Brown corpus, for example, results in a much larger tagset of 85 tags instead of 45, but the additional 40 complex tags cover a total of only 121 word instances out of the million word corpus.

NLP UNIT-2: English Word Classes, POS Tags, Types of POS Tagging

Most tagging algorithms assume a process of tokenization has been applied to the tags. An additional role for tokenization is in word splitting. The Penn Treebank and the British National Corpus split contractions and the *'s*-genitive from their stems:

would/MD n't/RB

children/NNS 's/POS

Indeed, the special Treebank tag POS is used only for the morpheme *'s* which must be segmented off during tokenization. Another tokenization issue concerns multi-part words. The Treebank tagset assumes that tokenization of words like *New York* is done at whitespace. The phrase *a New York City firm* is tagged in Treebank notation as five separate words: *a/DT New/NNP York/NNP City/NNP firm/NN*. The C5 tagset, by contrast, allow prepositions like "*in terms of*" to be treated as a single word by adding numbers to each tag, as in *in/II31 terms/II32 of/II33*.

Unknown Words

All the tagging algorithms we have discussed require a dictionary that lists the possible parts-of-speech of every word. But the largest dictionary will still not contain every possible word. Proper names and acronyms are created very often, and even new common nouns and verbs enter the language at a surprising rate. Therefore in order to build a complete tagger we cannot always use a dictionary to give us $p(wi|ti)$. We need some method for guessing the tag of an unknown word.

The simplest possible unknown-word algorithm is to pretend that each unknown word is ambiguous among all possible tags, with equal probability. Then the tagger must rely solely on the contextual POS-trigrams to suggest the proper tag. A slightly more complex algorithm is based on the idea that the probability distribution of tags over unknown words is very similar to the distribution of tags over words that occurred only once in a training set, an idea that was suggested by both Baayen and Sproat (1996) and Dermatas and Kokkinakis (1995). These words that only occur once are known as **hapax legomena** (singular **hapax legomenon**). For example, unknown words and *hapax legomena* are similar in that they are both most likely to be nouns, followed by verbs, but are very unlikely to be determiners or interjections. Thus the likelihood $P(wi|ti)$ for an unknown word is determined by the average of the distribution over all singleton words in the training set. This idea of using "things we've seen once" as an estimator for "things we've never seen" will prove useful in the Good- Turing algorithm.

Most unknown-word algorithms, however, make use of a much more powerful source of information: the morphology of the words. For example, words that end in *-s* are likely to be plural nouns (NNS), words ending with *-ed* tend to be past participles (VBN), words ending with *able* tend to be adjectives (JJ), and so on. Even if we've never seen a word, we can use facts about its morphological form to guess its part-of-speech. Besides morphological knowledge, orthographic information can be very helpful. For example words starting with capital letters are likely to be proper nouns (NP). The presence of a hyphen is also a useful feature; hyphenated words in the Treebank version of Brown are most likely to be adjectives (JJ). This prevalence of JJs is caused by the labeling instructions for the Treebank, which specified that prenominal modifiers should be labeled as JJ if they contained a hyphen.

A Non HMM-Based approach to unknown word detection was that of Brill (1995) using the TBL algorithm, where the allowable templates were defined orthographically (the first *N* letters of the words, the last *N* letters of the word, etc.). Most recent approaches to unknown word handling, however, combine these features in a third way: by using maximum entropy (**MaxEnt**) models such as the **Maximum Entropy Markov Model** (**MEMM**) first introduced by Ratnaparkhi (1996) and McCallum et al. (2000), and which we will study in Ch. 6. The maximum entropy approach is one a family of loglinear approaches to classification in which many features are computed for the word to be tagged, and all the features are combined in a model based on multinomial logistic regression. The unknown word model in the tagger of

Toutanova et al. (2003) uses a feature set extended from Ratnaparkhi (1996), in which each feature represents a property of a word, including features like:

word contains a number

word contains an upper-case letter

word contains a hyphen

word is all upper-case

word contains a particular prefix (from the set of all prefixes of length ≤ 4)

word contains a particular suffix (from the set of all prefixes of length ≤ 4)

word is upper-case and has a digit and a dash (like *CFC-12*)

word is upper-case and followed within 3 word by Co., Inc., etc