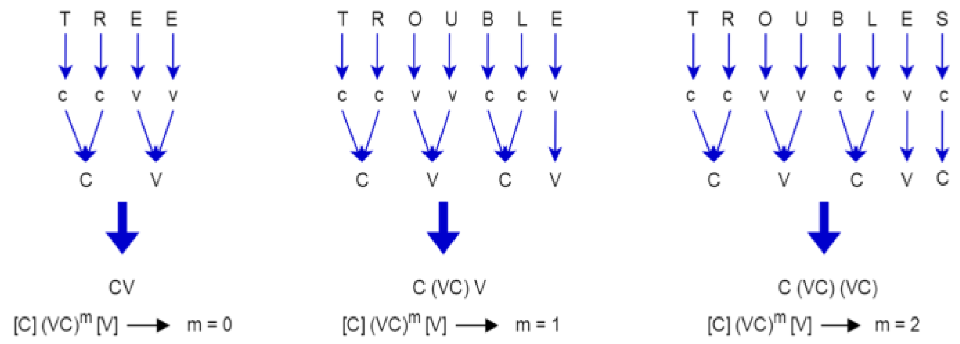
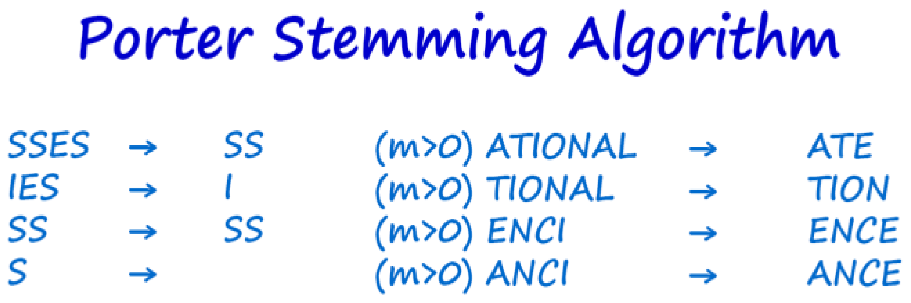
PORTER STEMMING ALGORITHM – BASIC INTRO




Porter Stemming Algorithm

SSES	→	SS	(m>0) ATIONAL	→	ATE
IES	→	I	(m>0) TIONAL	→	TION
SS	→	SS	(m>0) ENCI	→	ENCE
S	→		(m>0) ANCI	→	ANCE

In linguistics (study of language and its structure), a **stem** is part of a word, that is common to all of its inflected variants.

- CONNECT
- CONNECTED
- CONNECTION
- CONNECTING

Above words are **inflected variants** of CONNECT. Hence, CONNECT is a stem. To this stem we can add different suffixes to form different words.
The process of reducing such inflected (or sometimes derived) words to their word stem is known as **Stemming**. For example, CONNECTED, CONNECTION and CONNECTING can be reduced to the stem CONNECT.
The **Porter Stemming algorithm** (or **Porter Stemmer**) is used to **remove the suffixes from an English word and obtain its stem** which becomes very useful in the field of **Information Retrieval (IR)**. This process reduces the number of terms kept by an IR system which will be advantageous both in terms of space and time complexity. This algorithm was developed by a British Computer Scientist named **Martin F. Porter**. You can visit the **official home page** of the Porter stemming algorithm for further information.
First, a few terms and expressions will be introduced, which will be helpful for the ease of explanation.

Consonants and Vowels

A **consonant** is a letter other than the vowels and other than a letter "Y" preceded by a consonant. So in "TOY" the consonants are "T" and "Y", and in "SYZYGY" they are "S", "Z" and "G".
If a letter is not a consonant it is a **vowel**.
A consonant will be denoted by **c** and a vowel by **v**.
A list of one or more consecutive consonants (ccc…) will be denoted by **C**, and a list of one or more consecutive vowels (vvv…) will be denoted by **V.** Any word, or part of a word, therefore has one of the four forms given below.

- **CVCV … C** → collection, management
- **CVCV … V** → conclude, revise
- **VCVC … C** → entertainment, illumination

- **VCVC ... V** → illustrate, abundance

All of these forms can be represented using a single form as,

[C]VCVC ... [V]

Here the square brackets denote arbitrary presence of consonants or vowels.

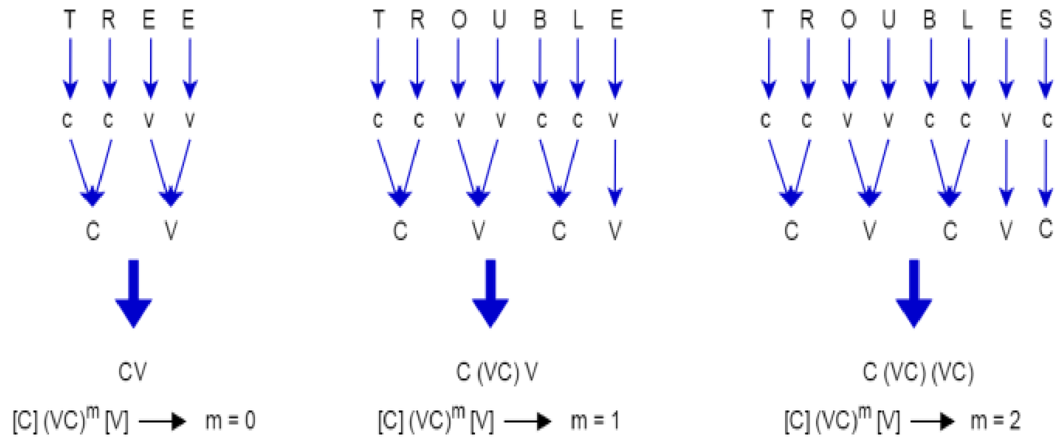
$(VC)^m$ denotes VC repeated m times. So the above expression can be written as,

$[C](VC)^m[V]$

What is m?

The value **m** found in the above expression is called the **measure** of any word or word part when represented in the form **$[C](VC)^m[V]$**. Here are some examples for different values of m:

- m=0 → TREE, TR, EE, Y, BY
- m=1 → TROUBLE, OATS, TREES, IVY
- m=2 → TROUBLES, PRIVATE, OATEN, ROBBERY



Rules

The rules for replacing (or removing) a suffix will be given in the form as shown below.

(condition) S1 → S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m in regard to the stem before S1.

(m > 1) EMENT →

Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

Conditions

The conditions may contain the following:

- *S – the stem ends with S (and similarly for the other letters)
- *v* – the stem contains a vowel
- *d – the stem ends with a double consonant (e.g. -TT, -SS)
- *o – the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP)

And the condition part may also contain expressions with and, or and not.

(m>1 and (*S or *T)) tests for a stem with m>1 ending in S or T.

(*d and not (*L or *S or *Z)) tests for a stem ending with a double consonant and does not end with letters L, S or Z.

How rules are obeyed?

In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching S1 for the given word. For example, with the following rules,

1. SSES → SS
2. IES → I
3. SS → SS
4. S →

(Here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally CARESS maps to CARESS (since S1="SS") and CARES to CARE (since S1="S").

The Algorithm

Step 1a

1. SSES → SS
2. IES → I
3. SS → SS
4. S →

Step 1b

1. (m>0) EED → EE
2. (*v*) ED →
3. (*v*) ING →

If the second or third of the rules in Step 1b is successful, the following is performed.

1. AT → ATE
2. BL → BLE
3. IZ → IZE
4. (*d and not (*L or *S or *Z)) → single letter
5. (m=1 and *o) → E

Step 1c

1. (*v*) Y → I

Step 2

1. (m>0) ATIONAL → ATE
2. (m>0) TIONAL → TION
3. (m>0) ENCI → ENCE
4. (m>0) ANCI → ANCE
5. (m>0) IZER → IZE
6. (m>0) ABLI → ABLE
7. (m>0) ALLI → AL
8. (m>0) ENTLI → ENT
9. (m>0) ELI → E
10. (m>0) OUSLI → OUS
11. (m>0) IZATION → IZE
12. (m>0) ATION → ATE
13. (m>0) ATOR → ATE

14. (m>0) ALISM	→	AL
15. (m>0) IVENESS	→	IVE
16. (m>0) FULNESS	→	FUL
17. (m>0) OUSNESS	→	OUS
18. (m>0) ALITI	→	AL
19. (m>0) IVITI	→	IVE
20. (m>0) BILITI	→	BLE

Step 3

1. (m>0) ICATE	→	IC
2. (m>0) ATIVE	→	
3. (m>0) ALIZE	→	AL
4. (m>0) ICITI	→	IC
5. (m>0) ICAL	→	IC
6. (m>0) FUL	→	
7. (m>0) NESS	→	

Step 4

1. (m>1) AL	→	
2. (m>1) ANCE	→	
3. (m>1) ENCE	→	
4. (m>1) ER	→	
5. (m>1) IC	→	
6. (m>1) ABLE	→	
7. (m>1) IBLE	→	
8. (m>1) ANT	→	
9. (m>1) EMENT	→	
10. (m>1) MENT	→	
11. (m>1) ENT	→	
12. (m>1 and (*S or *T)) ION	→	
13. (m>1) OU	→	
14. (m>1) ISM	→	
15. (m>1) ATE	→	
16. (m>1) ITI	→	
17. (m>1) OUS	→	
18. (m>1) IVE	→	
19. (m>1) IZE	→	

Step 5a

1. (m>1) E	→	
2. (m=1 and not *o) E	→	

Step 5b

1. (m > 1 and *d and *L)	→	single letter

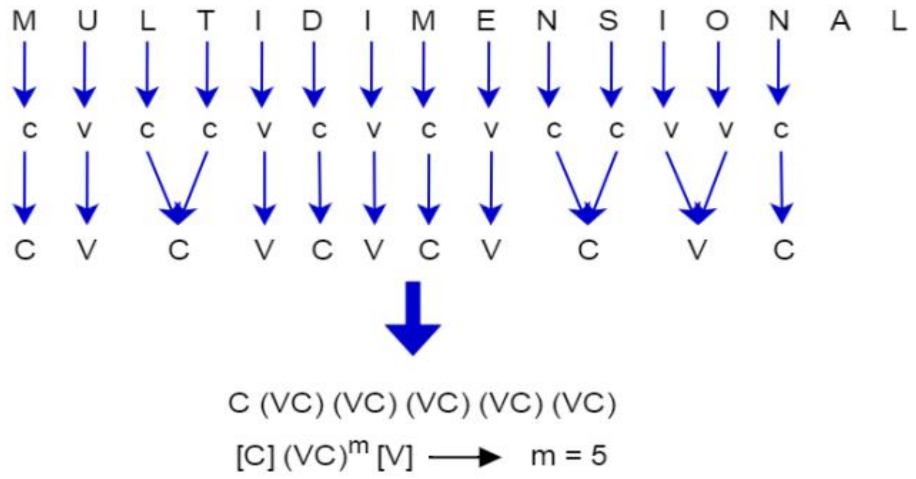
For each word you input to the algorithm, all the steps from 1 to 5 will be executed and the output will be produced at the end.

Example Inputs

Let’s consider a few example inputs and check what will be their stem outputs.

Example 1

In the first example, we input the word **MULTIDIMENSIONAL** to the Porter Stemming algorithm. Let's see what happens as the word goes through steps 1 to 5.



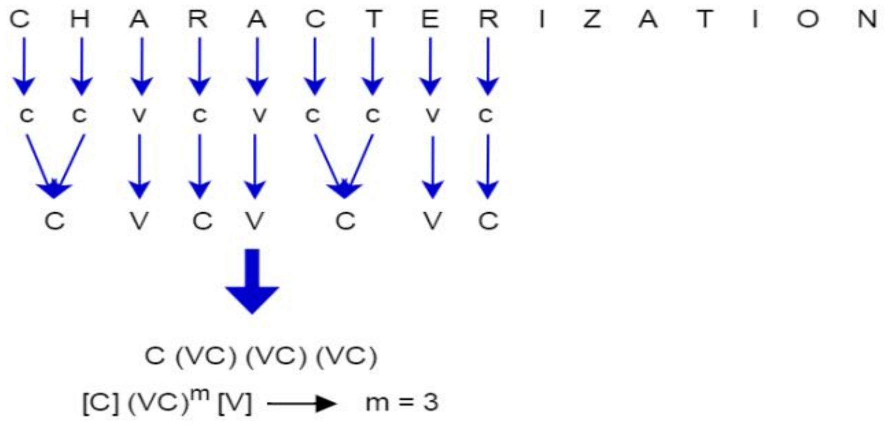
M U L T I D I M E N S I O N A L
c v c c v c v c v c c v v c
C V C V C V C V C V C
C (VC) (VC) (VC) (VC) (VC)
[C] (VC)m [V]
m = 5

- The suffix will not match any of the cases found in steps 1, 2 and 3.
- Then it comes to step 4.
- The stem of the word has m > 1 (since m = 5) and ends with "**AL**".
- Hence in step 4, "**AL**" is deleted (replaced with null).
- Calling step 5 will not change the stem further.
- Finally the output will be **MULTIDIMENSION**.

MULTIDIMENSIONAL → **MULTIDIMENSION**

Example 2

In the second example, we input the word **CHARACTERIZATION** to the Porter Stemming algorithm. Let's see what happens as the word goes through steps 1 to 5.



- The suffix will not match any of the cases found in step 1.
- So it will move to step 2.
- The stem of the word has m > 0 (since m = 3) and ends with "**IZATION**".
- Hence in step 2, "**IZATION**" will be replaced with "**IZE**".
- Then the new stem will be **CHARACTERIZE**.
- Step 3 will not match any of the suffixes and hence will move to step 4.
- Now m > 1 (since m = 3) and the stem ends with **"IZE"**.
- So in step 4, **"IZE"** will be deleted (replaced with null).
- No change will happen to the stem in other steps.
- Finally the output will be **CHARACTER**.

CHARACTERIZATION → CHARACTERIZE → **CHARACTER**