INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

OPEN ACCESS
Received: 27-03-2023
Accepted: 17-08-2023
Published: 30-09-2023

Citation: Rihan J, Astikar S (2023) Enhancement in Stemmer Design: Natural Language Semantics Perspective. Indian Journal of Science and Technology 16(37): 3050-3063. https://doi.org/10.17485/IJST/v16i37.711

* **Corresponding author**.
rihan.jasleen@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Enhancement in Stemmer Design: Natural Language Semantics Perspective

Jasleen Rihan[1]*, Shridhar Astikar[2]

1 Assistant Professor, Department of Humanities, Shri Ramdeobaba College of Engineering and Management, Nagpur
2 Research Scholar, Shri Ramdeobaba College of Engineering and Management, Nagpur

Abstract

Objective: To enhance the performance and accuracy of the stemming process. **Method:** The Porters stemmer is used conventionally for removing common morphological and inflectional endings (suffixes) from the words in the English language. It uses a set of pre-defined rules that are less complex when compared to other existing stemmers. We have identified several imprecisions encountered during the stemming process and proposed solutions to remove and invalidate the same. **Findings:** The experiment was performed on a set of 762 words starting with characters “a”, “b”, and “c”. It was found that out of 762 words used for system validation and testing, the results of 355 words were different when stemmed with MPS [Modified Porter Stemmer], and the remaining 407 words resulted in the same stemmed word after using both stemmers. The Modified Porter Stemmer presented in the current paper with Python implementation has given better results for 46% of words. **Novelty:** This paper highlights the encountered errors while using the algorithm and provides solutions to enhance the performance and accuracy of the stemming process. The designed stemmer is named “Modified Porter Stemmer” [MPS].

Keywords: Natural Language Processing; Stemmer; Porter’s Stemmer; Enhancement; Stemming Process

1 Introduction

Natural language texts contain numerous variants of a basic word. The most common variant of a word is the inflectional variant, which includes the usage of affixes to create a new word from the same base word called the root (eg. Medical, Medicine, Medicinal, Media). Inflection is the process that modifies a word and classifies it into grammatical categories, such as case, tense, gender, number, etc [(1)]. Thus, although a word may exist in several inflected forms, having multiple inflected forms within the same text document adds redundancy to the Natural Language Processing (NLP) process.

The NLP process uses the stemming technique to reduce the inflection in words, supplementing the processing of documents for text normalization. This technique reduces words to their basic form or stem, which may or may not be a valid and admissible word in the language. For instance, the stem of the words 'cloth', 'clothed', 'cloths', and 'clothing' is 'cloth'. But the root for 'pierce', 'pierced', 'pierces', and 'piercing' is 'pierc', which is non-existent in the English language. Such variations in a text result in data ambiguity and redundancy when developing machine learning models, ultimately leading to ineffectiveness and zero utility[2]. To build a robust learning model, it is imperative to remove such repetitions and ambiguities that occur while stemming to change the word into its root or base form.

There are several types of stemming algorithms in Python NLTK, and they differ with respect to their performance and precision. The most commonly used is the Porter Algorithm or Porter Stemmer, invented by Martin Porter in 1980. The stemmer is known for its rapid speed and ease of use. It mainly focuses on removing the inflectional endings of words to disintegrate them into common forms[3]. The output variants may often not be meaningful words. Hence, the algorithm continues to display several potential imprecisions and drawbacks. These drawbacks are discussed further in the later sections[4].

The stemming algorithm designed is based on context sensitivity of text. The term "context stripping" is used to find out the correct stem word for improving the accuracy of the word. The major challenge illustrated in the research work is the reduction in the accuracy if the suffix "itive" and "iti" is found in the text.[5,6].

It is found that the stemming process accuracy can be increased by modifying the rules of Porter Stemmer. In[7] an idea related to application of rules based algorithms on the word morphology and its instance is illustrated as future work. The presented research work illustrates the use of one such idea based on stem of the word and the test cases are executed.

In[8] a description of framework for implementing the stemmer in designing of search engines is presented. This research article indicated that if the stemmer is accurate the search engine accuracy is improved. The improvement is majorly found if stemming is applied at the run time with a gradient descent approach. The major challenge described in the paper is related to the dependency of accuracy on the training data set for creating a small granule of text for the stemming process.

In[9] the description of morphological derivatives and its use in the stemming process is illustrated. The BiLSTM model is used in the research work by authors to demonstrate the relationship between stemming and morphological derivative. The stemming model integrates the sentence context and character features. The experiments are carried out on limited data, which is further expanded in the presented research work for the improvement of accuracy.

In[10] the description of role of the stemming algorithm and its necessity in improving the accuracy of machine translation in a real-time environment is highlighted. The stemmer design is important feature of Natural Language Processing domain and its implementation and integration in any application related to machine translation is necessary for reducing the time for translation.

In[11] proposed ensemble model comprises three hybrid deep learning models which are a combination of Robustly optimized Bidirectional Encoder Representations from Transformers approach (RoBERTa), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM) and Gated Recurrent Unit (GRU). In all these three models, the role of "stemmer" and its implementation has played a major role in controlling the accuracy of the approach. The major challenge for future work illustrated in the research paper is related to unavailability of modified rules of "porter stemmer" and thus the accuracy of the system has not reached to the desired threshold.

In[12] the demonstration of the role of NLP in intrusion detection is presented. The role of accuracy and speedup techniques and their importance in intrusion detection systems has significant impact in the current scenario. The accuracy and speed depends on the stemmer results. The demonstrated framework makes use of a porter stemmer, which if modified can be useful in generating better results and accuracy.

Numerous attempts have been made to improve the structure of Porters algorithm to enhance its accuracy and efficacy. One such system showed all words produced from the proposed algorithm had a meaning, even while assessing lengthy documents. However, the major drawback of this approach was that the database did not have all the stems or English words to deliver 100% accurate results. In another study[12], researchers developed an improved model of Porter stemmer to evaluate the error counting method. Although the results demonstrated improved accuracy, this study did not assess the system in an information retrieval context. On the other hand researchers developed a modified version of Porters stemming system that reduced the error system from 21% for the original Porter system to an incredible value of 3%. However, the algorithm does generate unintelligible stems, such as seriou for serious.

Over-stemming and under-stemming are the two significant issues with Porters stemming algorithm. Hence, the current work is based on distinct rules to handle generic English words along with specific suffixes, with certain fixed defined conditions. The designed system checks each rule of stemming sequentially and removes the suffix or modification in the English word to avoid over-stemming and under-stemming.

- **Errors in Stemming**

The two identified errors in the stemming process are[4]

a) Over-stemming
b) Under-stemming

While using the stemming algorithm in natural language processing, larger words are chopped off to derive their roots. When two words are formed from the same root that originally have different stems, then it is referred to as Over-stemming. Often this is also termed as false-positives. In over-stemming, the stemmer produces a root form that is an invalid word or is an incorrect root form of a word[5]. This is usually a result of the insistent functioning of the stemmer while removing suffixes from the words without considering the lexical or contextual meaning of the word.

Over-stemming can lead to a loss of meaning and hamper the readability and understanding of a given text. For example, the word 'amusing', 'amusement', and 'amused' may be stemmed from 'amus', which is not a valid word, and does not convey a meaning similar to the original word[6]. Similarly, the word 'sleeping' will get stemmed from 'sleep', which is the base form of the word, yet it fails to convey the meaning of the original word.

- **Under-stemming**

occurs when two words are stemmed from the same root that are not of different stems. Under-stemming is often interpreted as false-negatives. In this process, a stemmer fails to produce the correct root form of a word or does not reduce a word to its actual base form. This is a result of a considerably less aggressive functioning of a stemmer while removing suffixes or when it is unable to perform a task for a specific language.

Under-stemming can lead to a loss of information and increased difficulty in analyzing the text. For example, the word 'data' may be reduced to 'dat' while the word 'datum' may be reduced to 'datu'. Both the words 'data' and 'datum' have the same root, yet they form two separate words.

Over-stemming and Under-stemming can be reduced and avoided by using the appropriate stemmer to perform a task for a particular language. The use of lemmatizer that reduces a word to 'lemma'- a word that is valid owing to its existence in a given language can help make the stemming process less susceptible to errors[7]. Various other techniques, such assemantic role labeling, sentiment analysis, context-based information, etc., can also be used to understand the context of the text, thus making the stemming process precise and effective.

- **Porter Stemmer**

The Porter Stemmer was developed and introduced by Martin Porter at Cambridge University in 1980. It was first published in Porter M.F and revised by Sparck, Karen, and Peter[6], by including stemming rules for improving accuracy. The researchers described stemming as a process of removing commoner morphological and inflexional suffixes from English words. The primary application of Stemming is in the domain of information retrieval and text mining. The most common suffixes found during the stemming process are "gerunds," "plurals," replacing words ending with "ator" etc.

The presented research work is based on the formation of rules to handle generic English words, along with specific suffixes, with certain fixed defined conditions. The designed code checks each rule of stemming sequentially and finally removes suffixes or modification in the English word.

- **Drawbacks of Porter Stemmer**

In the domain of information retrieval, vector of words are used in different senses, for example, Connect, Connection, Connections, Connecting, Connected. Upon presenting the vector of words to the stemmer, it is expected to get the final word as Connect. But if the same rule is applied to a different set of words like Relations, Relating, then the anticipated word is "Relate." This is not possible in the Porter Stemmer[1,2].

The primary drawbacks of the Porter Stemmer are:

1. The Stemming process is rule-based and fixed.[3]
2. The Stemming process is not based on the context of a word within the sentence.
3. The Stemming process does not consider the phonetics of a word and its resultant outcome after stemming.[4]
4. The Stemmer does not guarantee the true sense of word resulting after the stemming process.

- **Implementation**

Python Code for the functions used in solutions for errors in Porter Stemmer.

Function to check whether at a given index a letter is a consonant

Description: This function will check whether the letter at a given index in the word is a consonant or a vowel[2,3].

```
def isCons(word,index):
vowels = 'aeiou'
letter = word[index].lower()
if letter in vowels:
return False
elif letter == 'y' and (index != 0 and word[index-1].lower() not in vowels):
return False
else:
return True
```

For example: P A S T E → will result in → C V C C V →C V C V

 # Function to count Vowel-Consonant Pairs

Description: As the Vowel-Consonant pair is used in stemming process, the function described will count the number of Vowel-Consonant pairs in a given word.

```
def vcPair(send):
count = 0
k = len(send) - 1
for i in range (k):
if not isCons(send, i) and isCons(send, i+1):
count += 1
return count
```

For example: P A S T E → will result in → C V C C V → C V C V → VC pair = 1 [m=1]

 # Function to check whether the string is a Vowel-Consonant series

```
def vcSeries(send):
p = len(send) - 1
if isCons(send, p):
while(p > 0):
if isCons(send, p) and not isCons(send, p-1):
p -= 2
else: return False
if p < 0 or (p == 0 and isCons(send, 0)):
return True
else: return False
else:
while(p > 0):
if not isCons(send, p) and isCons(send, p-1):
p -= 2
else: return False
if p < 0 or (p == 0 and not isCons(send, 0)):
return True
else: return False
```

The following section describes the different types of errors identified at the language level while using the Porter Stemmer for the purpose of stemming. These errors are a result of a fixed approach followed during the stemming process. The section also describes the suggested modifications using relevant examples in each category of error.

Errors and suggested Modifications:

Error #1:

The error is caused due to deletion of letter "e" for the words in which m=1 [VC pair] and ends with two consonants. [3,4]

G: range, paste [VC pair = 1]

Paste → past Range → rang

Past → past Rang → rang

The proposed solution will be used to stem/remove the "suffix" from the word without changing its meaning[4,5].

Suggested Solution for Error#1

A function is created to keep the letter "e" at the end of words if words ends with TWO consonants and m=1. For example: Paste=Paste.

The solution creates another sub problem for the words ending with "ches" or "shes". For such suffix, the function removes: "es". For example: Beaches=Beach and Bushes = Bush

 # Solution 1 [Python Code]

```
def endsWithE(word):
if word[-1] == 'e':
send = word[:-1]
p = len(send) - 1
if vcPair(send) == 1:
if isCons(word, 0) and isCons(word, p) and isCons(word, p-1):
return word
else:
return word[:-1]
elif word.lower().endswith("ches") or word.lower().endswith("shes"):
return word[:-2]
else:
print("Not in case")
```

Output Generated by Modified Rule

Range → Range
Paste → Paste
Beaches → Beach
Bushes → Bush
Loathe → Loathe
Bottle → Bottle
Wooshes → Woosh

Error #2;

In words ending with "is", the letter "s" is removed in the process [7].

e.g. His, appendicitis Conjunctivitis → conjunct Conjunct → conjunct

 # Solution 2 [Python Code]

```
def endsWithIS(word):
if word.lower().endswith("is"):
return word
else:
print("Not in case")
```

Output Generated by Modified Rule

Conjunctivitis → Conjunctivitis
Basis → Basis

Error #3:

The error is caused when the words with ending with "yed" and "ying" results in same stem.

Dying → dy (impregnate with dye)
Dyed → dy (passes away)

Suggested Solution for Error#3 [7]

To prevent words ending with "ying" and "yed" producing same stem, but initially having different meaning, the suffix "ying" is set to "i" if word starting with consonant and vowel.

For example: Dying = CCVCC = CVC, then Dying = Di

 # Solution 3: [Python Code]

```
def endsWithYING(word):
if word.lower().endswith("ying"):
send = word[:-4]
if vcPair(send) == 0 and isCons(word, 0) and not isCons(word, 1):
return word[:-4] + 'i'
elif vcPair(send) == 0 and isCons(word, 0) and len(send) == 1:
```

```
return word[:-4] + 'i'
else: return word
elif word.lower().endswith("yed"): return word[:-2]
else:
print("Not in case")
```

Output

Dying → Di
Crying → Crying
Disqualifying → Disqualifying
Dyed → Dy
Swayed → Sway

Error #4:

The error is caused with m=2 and words ending with series of CVCV with "ic" or "ical" is removed.[6,7]

e.g. Politic, generic; Satirical → satir; Satiric → satir; Satire → satir

Suggested Solution for Error#4[7]

For such words the deletion and addition of new suffix is carried out based on characters existing in the suffix. For example: Political → remove "ical" and then add "ic": Resultant word=Politic.

 # Solution 4 [Python Code]

```
def endsWithIC(word):
if word.lower().endswith("ic"):
send = word[:-2]
print(vcPair(send))
if vcPair(send) == 2:
if vcSeries(send):
word = send[:-2] + "ica*"
return word[:-2]
print("op1")
else: return word[:-4]
else: return word[:-4]
elif word.lower().endswith("ical"):
send = word[:-4]
if vcPair(send) == 2:
if vcSeries(send):
word[:-2] + "ica*"
return word
else: return word[:-4]
else: return word[:-4]
else:
print("Not in case")
```

Output

Medical → Med
Satirical → Satirical
Political → Political
Pharmaceutical →Pharmaceut
Epic → Ep
Politic → Polit
Macrocyclic → Macrocycl

Error #5:

The error is caused due to removal of the suffix "ative" from all the words ending with it and having m=1 or m=2[9]

Alternative → altern   Generative → gener
Altern → altern   General → gener

Suggested Solution for Error#5

If the word ends with "ative" and m=2, replace it by "ate"

Generative → Generate
If m>2, then, it is removed
Authoritative → Authorit
If m=1, it is replaced by "at"
Combative → Combat

 # Solution 5 [Python Code]

```
def endsWithATIVE(word):
if word.lower().endswith("ative"):
send = word[:-5]
if vcPair(send) == 2:
word = word[:-5] + "ate"
return word
elif vcPair(send) == 1:
return word[:-5] + "at"
elif vcPair(send) > 2:
return word[:-5]
else:
return word
else:
print("Not in case")
```

Output

Native → Native   Combative → Combat   Generative → Generate
Collaborative → Collabor

Error #6:

This error is resultant of removal of suffix "ness" from the words where m=1 and also the words ends with CVC series.[10]

Witness → wit   Shyness → shy
Wit → wit   Shy → shy

Suggested Solution for Error#6

If the word ends with "ness" and m=1 and ends with CVC, then it is kept as it is. For example Witness →Witness, else it is removed.

 # Solution 6 [Python Code]

```
def endsWithNESS(word):
if word.lower().endswith("ness"):
send = word[:-4]
l = len(send) - 1
if vcPair(send) == 1 and isCons(word, l) and not isCons(word, l-1) and isCons(word, l-2):
return word
else:
return word[:-4]
else:
print("Not in case")
```

Output

Witness → Witness
Shyness → Shy
Setness → Set
Loneliness → Loneli

Error #7:

The error is caused due to removal of suffix "al" is removed from all words where m=2
e.g.Natural, animal, admiral
Imaginal → imagin   Admiral → admir
Imagine → imagin   Admire → admir

Suggested Solution for Error#7

If word ends with "iral", and m=2, then no change.

For example: Admiral → Admiral
If word ends with "al", and m=2, and it consist of series of CVCV, then "al" is removed
General = CVCVCVC → removal "al" → Gener
 # Solution 7 [Python Code]
```
def endsWithAL(word):
if word.lower().endswith("al"):
send = word[:-2]
if vcPair(send) == 2:
if word.lower().endswith("iral"):
return word
elif vcSeries(send):
return word[:-2]
else: return word
elif vcPair(send) > 1:
return word[:-2]
else: return word
else:
print("Not in case")
```
Output
Pal → Pal
Medal → Medal
General → Gener
Chiral → Chiral
Imaginal → Imaginal
Admiral → Admiral
Antiviral → Antivir
Error #8:
The error is resultant of removal of suffix "eer" is eliminated from all the words with m=2
For ex: engineer, privateer
Privateer → privat   Engineer → engin
Private → privat   Engine → engin
Suggested Solution for Error#8
If the word ends with "eer" and m=2, then only "r" is removed and consequently the last "e" is also removed.
Engineer → Engine
 # Solution 8 [Python Code]
```
def endsWithER(word):
if word.lower().endswith("er"):
send = word[:-3]
if word.lower().endswith("eer") and vcPair(send) == 2:
return word[:-2]
else:
return word
else:
print("Not in case")
```
Output
Cheer → Cheer
Pioneer → Pioneer
Engineer → Engine
Fiber → Fiber
Banner → Banner
Error #9:
The error is caused when, suffix "ible" is excluded from all words where m=2, start with a consonant and do not end with a series of consonant, vowel, consonant, vowel.[11]

Dispersible → dispers
Disperse → dispers

Suggested Solution for Error#9[7]

If the word ends with "ible" and m=2 and starts with consonant and ending with series of CVCV, then no change. For example: Responsible → Responsible, otherwise, it is removed: Reducible → Reduc.

 # Solution 9 [Python Code]

```
def endsWithIBLE(word):
if word.lower().endswith("ible"):
send = word[:-4]
if vcPair(send) == 2 and isCons(word, 0) and not vcSeries(send):
return word
else: return word[:-4]
else: print("Not in case")
```

Reducible → Reduc
Responsible → Responsible
Visible → Vis
Reprehensible → Reprehens

Error #10:

The error is found when the suffix "ance" is reduced from the word with m=2 and ends with series of CVCV.[12]

Conveyance → convey   Securance → secur
Conveyal → convey   Secure → secur

Suggested Solution for Error#10

If the word ends with "ance' and m=2, and consist of series of CVCV, then it is replaced by "e". For example: Severance → Severe. If not it is removed. ImportanceImport.

 # Solution 10 [Python Code]

```
def endsWithANCE(word):
if word.lower().endswith("ance"):
send = word[:-4]
p = len(send) - 1
if vcPair(send) == 2 and vcSeries(send):
word = send + 'e'
return word
else:
return word[:-4]
else:
print("Not in case")
```

Output

Distance → Dist
Importance → Import
Severe → Severe

Error #11:

The error is caused due to removal of suffix "ment" from the words or those ending with "iment" with m=2 and not ending with series of CVCV.[12]

Experiment → experi

Suggested Solution for Error#11[7]

If the word ends with "iment" and m=2, and not ending with CVCV, then no change. For example: Experiment → Experiment. If m>2, then it is removed. Accompaniment → Accompani

 # Solution 11 [Python Code]

```
def endsWithIMENT(word):
if word.lower().endswith("iment"):
send = word[:-5]
if vcPair(send) == 2 and not vcSeries(send):
return word
```

```
elif vcPair(send) > 1:
return word[:-4]
else: return word
elif word.lower().endswith(”ement”) or word.lower().endswith(”ment”):
return word
else:
print(”Not in case”)
```

Output

Regiment → Regi

Experiment → Experiment

Accompaniment → Accompani

Error #12:

The error is due to removal of “ion” from the words where m=2 and is not consonant, vowel, consonant, vowel without replacement.[12]

Secretion → secret

Secret → secret

Suggested Solution for Error#12

If word ending with “tion” and “m=2”, and not ending with series of CVCV, then it is replaced by “e”

Secretion → Secrete   Sedition → Sedit

 # Solution 12: [Python Code]

```
def endsWithION(word):
if word.lower().endswith(”ion”):
send = word[:-3]
if send[-1] == ’t’:
if vcPair(send) == 2 and not vcSeries(send):
word = send + ’e’
return word
elif vcPair(send) > 1:
return word[:-3]
else: return word
else:
print(”Not in case”)
```

Output

Station → Stat

Sedition → Sedit

Secretion → Secrete

Deposition → Deposit

Error #13:

The error is due to removal of suffix “nate” from the words where m=2 and ending with series of CVCV.

Designate → design

Design → design

Suggested Solution for Error#13[7]

If the word ends with “nate” or “ate” and m=2, also ends with series of CVCV, then no change. For example: Designate → Designate. If m>2, then it is removed. For example: Collaborate → Collabor. If m=1, then “at” is kept as it is. For example: Situate → Situat, if m=0, then no change.

 # Solution 13

```
def endsWithNATE(word):
if word.lower().endswith(”nate”):
send = word[:-4]
if vcPair(send) == 2 and vcSeries(send):
return word
else: return word[:-4]
elif word.lower().endswith(”ate”):
```

```
send = word[:-3]
if vcPair(send) == 2 and vcSeries(send):
return word
elif vcPair(send) > 1:
return word[:-3]
elif vcPair(send) == 1:
return word[:-1]
elif vcPair(send) == 0:
return word
else:
print("Not in case")
```

Output

Ate → Ate

Situate → Situat

Designate → Designate

Inordinate → Inordi   Collaborate → Collabor

Error #14:

This error is caused due to elminiation of suffix "oze" from the words having m=2 and starting with consonant. The words ends with series of CVCV.

Colonize → colon   Customize → custom

Colon → colon   Custom → custom

Suggested Solution for Error#14[7]

If the word ends with "ize" and m=2, and starting with "C" and ending with CVCV series, then no change. For example: Colonize → Colonize. If m>1, then it is removed. For example: Aerosolize → Aerosol [VVCVCVCVCV]

 # Solution 14 [Python Code]

```
def endsWithIZE(word):
if word.lower().endswith("ize"):
send = word[:-3]
if vcPair(send) == 2 and isCons(send, 0) and vcSeries(send):
return word
elif vcPair(send) > 1:
return word[:-3]
else : return word
else:
print("Not in case")
```

Output

Belize → Bel

Colonize → Colonize

Immunize → Immun

Aerosolize → Aerosol

Error #15:

This error is caused due to deletion of the suffix "itive" from the word with m=1. The word starts with a consonant and ends with a series of CVCV.[11,12]

Positive → posit

Position → posit

Suggested Solution for Error#15[7]

If the word ends with "itive" and m=1, starts with "C" and ends with series CVCV, then no change. For example: Positive → Positive, if m>1, then it is removed. For example: AcquistiveAcquisit

 # Solution 15 [Python Code]

```
def endsWithITIVE(word):
if word.lower().endswith("itive"):
send = word[:-5]
if vcPair(send) == 1 and isCons(send, 0) and vcSeries(send):
```

```
return word
elif vcPair(send) > 1:
return word[:-3]
else: return word
else:
print("Not in Case")
```

Output

Positive → Positive
Additive → Addit
Acquisitive → Acquisit
Competitive → Competit

Error #16:

This error is caused due to the removal of the suffix "iti" from the word with m=2. The word starts with a vowel and ends with a series of CVCV.

Ameniti → amen   Amen → amen

The same error also exists with m=3, for words starting with a vowel and not ending with a series of CVCV.

Universiti → univers   Universe → univers

Suggested Solution for Error#16 [7]

If the word ends with "iti" and m=2, and starts with "V" and ends with series of CVCV, then no change. For example: Amenity → Ameniti. If m=3, still no change. For example: Universiti → Universiti. If m>1, then it is removed. For example: Minority → Minor.

 # Solution 16 [Python Code]

```
def endsWithITI(word):
if word.lower().endswith("iti"):
send = word[:-3]
p = len(send) - 1
if vcPair(send) == 2 or vcPair(send) == 3 and isCons(send, 0) and vcSeries(send):
return word
elif vcPair(send) > 1:
return word[:-3]
else: return word
else:
print("Not in Case")
```

Output

Ameniti → Ameniti
Graffiti → Graff
Universiti → Univers

2 Experiment

A sample case study is presented below, illustrating the paragraph text used for testing the performance of Modified Porter Stemmer. The resultant table presents output with Porter2 and Modified Porter Stemmer, with correct/incorrect status. [1,2]

Sample I:

Jane was a **witness** to a terrifying robbery that took place on the private beach resort she worked in. It was an artistically constructed property situated alongside one of the most famous **beaches** in Miami. The resort was a luxurious property with not many guests during the monsoon. On checking, it was found that the robber had taken off with a **range** of expensive beach items, including umbrellas, a beach wagon, coasters, and snorkeling gear. Jane thought of herself to be **responsible** and decided to take action to prevent any such unfortunate incident in the future. She decided to **designate** a team of security personnel to ensure the **secure conveyance** of all the beach items. The security worked as per the instructions to implement the new safety measures to prevent such thefts. Jane also decided to create an **alternative range** of beach equipment that was less expensive but still of good quality. The new range of beach equipment included an assortment of chairs, umbrellas, and snorkeling gear, all of which were **competitively** priced. Jane also added a range of **amenities,** including a free water **bottle** and sunscreen, to make the beach experience even more **recreational.**

Table 1. Output Table

Word	Porter2	Modified Porter	Suffix	Status	P/MPS
Witness	Wit	Witness	ness	Different	MPS
Beaches	Beach	Beach	es	Same	P/MPS
Range	Rang	Range	e	Different	MPS
Responsible	Respons	Responsible	ible	Different	MPS
Designate	Design	Designate	ate	Different	MPS
Secure	Secur	Secur	e	Same	IC
Conveyance	Convey	Convey	ance	Same	P/MPS
Alternative	Altern	Alternate	ative	Different	MPS
Competitively	Competit	Competitively	Not in Case	Different	X
Amenities	Amen	Ameniti	es	Different	IC
Bottle	Bottl	Bottle	e	Different	MPS
Recreational	Recreat	Recreation	al	Different	MPS

The abbreviation used in P/MPS column are:
MPS: Correct results by Modified Porter Stemmer
P/MPS: Correct results by Porter2 and Modified Porter Stemmer
IC: Incorrect results by both Stemmer
X: The word not understood by the Stemmer.
Observation: The modified porter stemmer shows substantial accuracy over the Porter2 stemmer.

3 Results and Discussion

The experiment was carried out on a set of 762 words starting with characters "a," "b," and "c." The words and results of stemming for system comparison are taken from the source http://snowball.tartarus.org/. It was found that out of 762 words used for system validation and testing, the results of 355 words were different when stemmed with MPS [Modified Porter Stemmer], and the remaining 407 words resulted in the same stemmed word after using both stemmers.[3,4]

The Modified Porter Stemmer presented in the paper with Python implementation has given better results for 46% of words.

The term **"Word Stemming Factor"** is computed by finding the ratio of the number of words stemmed to the total number of words. For the presented work, the Word Stemming Factor is 79% with Modified Porter Stemmer and 64% with Porter2 stemmer.[4–6]

The term **"Correctly Stemmed Words"** is computed by finding the ratio of the number of words stemmed and found correct to the total number of words. For the presented work, the Correctly Stemmed Words is 46% with Modified Porter Stemmer and 39% with Porter2 stemmer.[4–6]

4 Conclusion

The Modified Porter Stemmer [MPS] is more suitable, as the stemming words generated after the process are more meaningful and relevant. If the Porter or Porter2 Stemmer is used, the suffix of the word is completely deleted[2]. This results in errors during the Machine Translation process. The Porter Stemmer is a baseline for designing stemmer, which can be modified based on the language used in the context of text.

References

1) Polus ME, Abbas T. Development for performance of Porter stemmer algorithm. *Eastern-European Journal of Enterprise Technologies*. 2021;1(2 (109)):6–13. Available from: https://doi.org/10.15587/1729-4061.2021.225362.
2) Asiri Y, Halawani HT, Alghamdi HM, Hamza SHA, Abdel-Khalek S, Mansour RF. Enhanced Seagull Optimization with Natural Language Processing Based Hate Speech Detection and Classification. *Applied Sciences*. 2022;12(16):8000. Available from: https://doi.org/10.3390/app12168000.
3) Khyani D, Siddhartha BS. An Interpretation of Lemmatization and Stemming in Natural Language Processing. . 2021. Available from: https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing.
4) Demidovich I, Shynkarenko V, Kuropiatnyk O, Kirichenko O. Processing Words Effectiveness Analysis in Solving the Natural Language Texts Authorship Determination Task. In: 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). IEEE. 2021;p. 48–51. Available from: https://doi.org/10.1109/CSIT52700.2021.9648829.
5) Jura J, Trnka P, Cejnek M. Using NLP to analyze requirements for Agriculture 4.0 applications. In: 2022 23rd International Carpathian Control Conference (ICCC). IEEE. 2022;p. 239–243. Available from: https://doi.org/10.1109/ICCC54292.2022.9805905.

6) Ramadhan A, Abdurachman E, Trisetyarso A, Zarlis M. Stemming Algorithm for Indonesian Language: A Scientometric View. IEEE Creative Communication and Innovative Technology. 2022. Available from: https://doi.org/10.1109/ICCIT55355.2022.10119050.
7) Pramana R, Debora, Subroto JJ, Gunawan AAS, Anderies. Systematic Literature Review of Stemming and Lemmatization Performance for Sentence Similarity. In: 2022 IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA). IEEE. 2022;p. 1–6. Available from: https://doi.org/10.1109/ICITDA55840.2022.9971451.
8) Şentürk F, Gunduz G. A framework for investigating search engines' stemming mechanisms: A case study on Bing. *Concurrency and Computation: Practice and Experience*. 2022;34(9). Available from: https://doi.org/10.1002/cpe.6562.
9) Imin G, Ablimit M, Yilahun H, Hamdulla A. A Character String-Based Stemming for Morphologically Derivative Languages. *Information*. 2022;13(4):170. Available from: http://dx.doi.org/10.3390/info13040170.
10) Khurana D, Koli A, Khatter K, Singh S. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*. 2023;82(3):3713–3744. Available from: https://doi.org/10.1007/s11042-022-13428-4.
11) Tan KL, Lee CP, Lim KM, Anbananthen KSM. Sentiment Analysis With Ensemble Hybrid Deep Learning Model. *IEEE Access*. 2022;10:103694–103704. Available from: https://doi.org/10.1109/ACCESS.2022.3210182.
12) Sworna ZT, Mousavi Z, Babar MA. NLP Methods in Host-based Intrusion Detection Systems: A Systematic Review and Future Directions". *Journal of Network and Computer Applications*. 2022. Available from: https://doi.org/10.48550/arXiv.2201.08066.