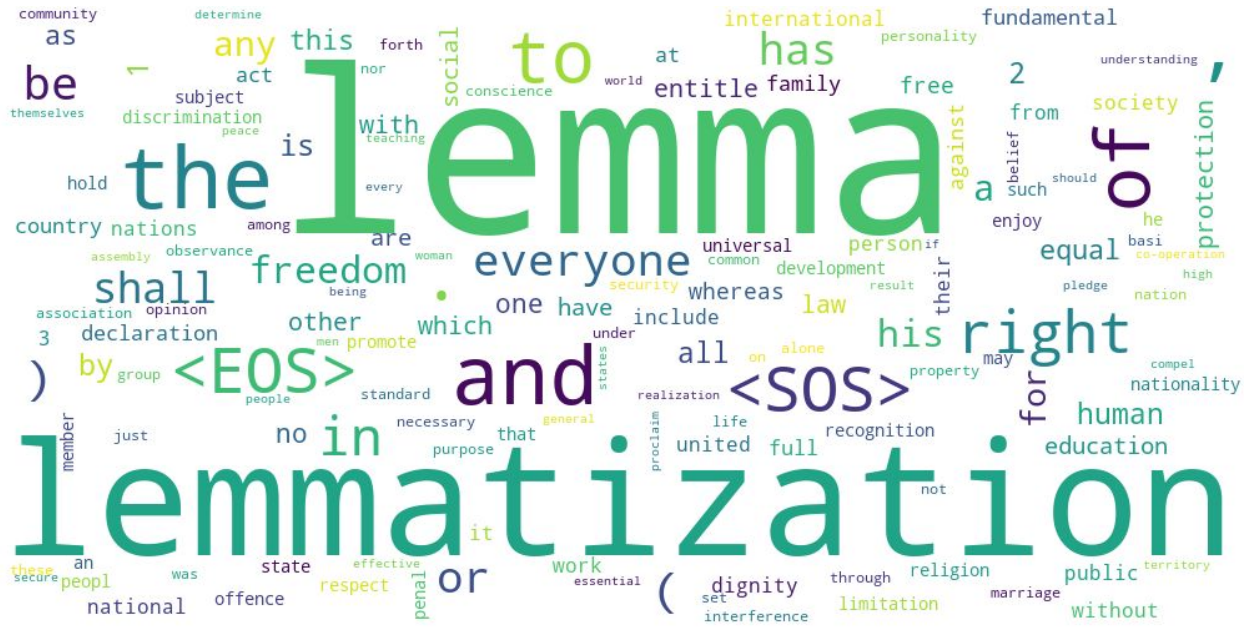

Natural Language Processing Assignment 4 - Lemmatization

Delhi Technological University - Dr. Seba Susan

Interactive Project on Jupyter Notebook 📓

Code on GitHub

1st September 2020

Anish Sachdeva
DTU/2K16/MC/13
Delhi Technological University

Index

Introduction.....2
Input.....3
Stemmed Output.....4-9
Lemmatized Output.....10-13
Analytics.....14-21
Discussion.....22-23
Bibliography.....24

Introduction

For grammatical reasons, documents are going to use different forms of a word, such as *organize*, *organizes*, and *organizing*. Additionally, there are families of derivationally related words with similar meanings, such as *democracy*, *democratic*, and *democratization*. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set.

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Stemming follows a heuristic approach to reducing the stem wherein the stemmer word will have similar character strings and may or may not be a valid dictionary word e.g.

happy --> happi

running --> run

The lemmatizer on the other hand doesn't follow a heuristic approach in reducing the word and will instead use a lookup table to refer to the various forms and meanings the word can possess and will also look up the surrounding context of the word to determine the correct POS (Part of Speech) Tag, which is then used to find the **lexeme** of the word. The lexeme is a valid english word and doesn't necessarily have to have a similar character (root) structure e.g.

better --> good

wolves --> wolf

Lemmatization has many practical real world use cases and is used very heavily as apre-processing step in many NLP tasks. It can help us in intent understanding or discourse resolution in chatbots and speech agents. If we wish to communicate with a user and the user is relaying typed instructions, the machine needs to actually understand the intention behind the command and this requires lemmatization so that synonyms such as better, best, good etc. can all be reduced down to → *good.*

Lemmatization also plays a big role in machine translation where we wish to compute the probability of a certain sequence of tokens or in sentiment analysis as well, wherein the word university and universe carry a very different sentiment, and their lemmatized forms will represent that whereas the stemmed form univer for both cases will not encapsulate the different sentiments.

We implement the lemmatizer using the Python **nltk** package and apply it on a resume and compare the output with a stemmed form of the same resume. Further Analytics and discussion give a deep dive into advantages, disadvantages and uses cases of lemmatization.

Input

As an input I use my resume, the same one which has been used in past assignments as well.

resume.txt [*see on GitHub*]

```
Anish Sachdeva

Software Developer + Clean Code Enthusiast


Phone : 8287428181

email : anish_@outlook.com

home : sandesh vihar, pitampura, new delhi - 110034

date of birth : 7th April 1998

languages : English, Hindi, French


Work Experience

What After College ( 4 months )

Delhi, India

Creating content to teach Core Java and Python with Data Structures and
Algorithms and giving online classes to students


Summer Research Fellow at University of Auckland ( 2 Months )

Auckland, New Zealand

Worked on Geometry of Mobius Transformations, Differential Geometry under Dr.
Pedram Hekmati at the Department of

Mathematics, University of Auckland


Software Developer at CERN ( 14 Months )

CERN, Geneva, Switzerland

Worked in the core Platforms team of the FAP-BC group. Part of an agile team of
developers that maintains and adds core

functionality to applications used internally at CERN by HR, Financial,
Administrative and other departments
```

including Scientific

Worked on legacy applications that comprise of single and some times multiple frameworks such as Java Spring, Boot,

Hibernate and Java EE. Also worked with Google Polymer 1.0 and JSP on the client side

Maintained CERN's Electronic Document Handing System application with >1M LOC that comprising of multiple frameworks

and created ~20 years ago. Worked on feature requests, support requests and incidents and also release cycles

Teaching Assistant (4 Months)

Coding Ninjas, Delhi

Served as the teaching assistant to Nucleus - Java with DS batch, under Mr. Ankur Kumar. Worked on creating course

content and quizzes for online platform of Coding Ninjas for Java. Helped students in core Data Structures and

Algorithms concepts in Java

Education

Delhi Technological University (2016 - 2021)

Bachelors of Technology Mathematics and Computing

CGPA: 9.2

The Heritage School Rohini (2004 - 2016)

Physics, Chemistry, Maths + Computer Science with English

Senior Secondary: 94.8%

Secondary: 9.8 CGPA

Technical Skills

Java + Algorithms and Data Structures

MEAN Stack Web Development

Python + Machine Learning

MATLAB + Octave

MySQL, PostgresSQL & MongoDB

Other Skills

MS Office, Adobe Photoshop, LaTeX + MiTeX

University Courses

Applied Mathematics I, II, III

Linear Algebra + Probability & Statistics + Stochastic Processes + Discrete Maths

Computer Organization & Architecture + Data Structures + Algorithm Design and Analysis + DBMS + OS

Computer Vision + NLP

Important Links

https://www.linkedin.com/in/anishsachdeva1998/

https://github.com/anishLearnsToCode

https://www.hackerrank.com/anishviewer

Stemmed Output

We now use the **PorterStemmer** created in Assignment 1 to stem our resume and we save the stemmed output as a string in the `resume_stemmed.p` file so that we can later see the output and also run analytics on it. We also save a text output under `resume_stemmed.txt` so that we can share the output.

stem_resume.py *[see on GitHub]*

```
from src.PorterStemmer import PorterStemmer
import pickle

stemmer = PorterStemmer()
resume_file = open('../assets/resume.txt', 'r')
resume = resume_file.read().lower()
resume_file.close()

resume_stemmed = stemmer.stem_document(resume)
pickle.dump(obj=resume_stemmed, file=open('../assets/resume_stemmed.p', 'wb'))

resume_stemmed_file = open('../assets/resume_stemmed.txt', 'w')
resume_stemmed_file.write(resume_stemmed)
resume_stemmed_file.close()
```

resume_stemmed.txt *[see on GitHub]*

anish sachdeva

softwar develop clean code enthusiast

phone 8287428181

email anish_ outlook com

home sandesh vihar pitampura new delhi 110034

date of birth 7th april 1998

languag english hindi french

work experi

what after colleg 4 month

delhi india

creat content to teach core java and python with data structur and algorithm and give onlin class to student

summer research fellow at univers of auckland 2 month

auckland new zealand

work on geometri of mobiu transform differenti geometri under dr pedram hekmati at the depart of

mathemat univers of auckland

softwar develop at cern 14 month

cern geneva switzerland

work in the core platform team of the fap bc group part of an agil team of develop that maintain and add core

function to applic us intern at cern by hr financi administr and other depart

includ scientif

work on legaci applic that compris of singl and some time multipl framework such a java spring boot

hibern and java ee also work with googl polym 1 0 and jsp on the client side

maintain cern s electron document hand system applic with 1m loc that compris of multipl framework

and creat 20 year ago work on featur request support request and incid and also releas cycl

teach assist 4 month

code ninja delhi

serv a the teach assist to nucleu java with d batch under mr ankur kumar work on creat cours

content and quizz for onlin platform of code ninja for java help student in core data structur and

algorithm concept in java

educ

delhi technolog univers 2016 2021

bachelor of technolog mathemat and comput

cgpa 9 2

the heritag school rohini 2004 2016

physic chemistri math comput scienc with english

senior secondari 94 8

secondari 9 8 cgpa

technic skill

java algorithm and data structur

mean stack web develop

python machin learn

matlab octav

mysql postgressql mongodb

other skill

m offic adob photoshop latex mitex

univers cours

appli mathemat i ii iii

linear algebra probabl statist stochast process discret math

comput organ architectur data structur algorithm design and analysi dbm o

comput vision nlp

import link

http www linkedin com in anishsachdeva1998

http github com anishlearnstocod

http www hackerrank com anishview

Lemmatized Output

We use the **`WordNetLemmatizer`** from the **`nltk.stem`** module part of the **`nltk`** package. We create our own **`Lemmatizer`** class that wraps the **`WordNetLemmatizer`** so that we can easily create instances of our **`Lemmatizer`** class which provides convenient API's to lemmatize words, sentences and documents.

Lemmatizer.py *[see on GitHub]*

```
import nltk
from nltk.stem import WordNetLemmatizer



class Lemmatizer:
  def __init__(self):
    self._lemmatizer = WordNetLemmatizer()
    self._tokenizer = nltk.RegexpTokenizer(r'\w+')

  def _tokenize(self, document: str) -> list:
    return self._tokenizer.tokenize(document)

  def lemmatize_word(self, word: str, pos=None) -> str:
    return self._lemmatizer.lemmatize(word, pos) if pos is not None else
self._lemmatizer.lemmatize(word)

  def lemmatize_sentence(self, sentence: str, pos=None) -> str:
    result = []
    for word in self._tokenize(sentence):
      if pos is not None:
        result.append(self.lemmatize_word(word, pos))
      else:
        result.append(self.lemmatize_word(word))
    return ' '.join(result)
```

```
def lemmatize_document(self, document: str) -> str:
    result = []
    for line in document.split('\n'):
        result.append(self.lemmatize_sentence(line))
    return '\n'.join(result)
```

We use the **`lemmatize_resume.py`** file to lemmatize the resume.txt file and save the output as a string in the **`resume_lemmatized.p`** pickle file for later analytics and resume_lemmatized.txt file for seeing the output.

resume_lemmatized.txt *[see on GitHub]*

```
anish sachdeva

software developer clean code enthusiast


phone 8287428181

email anish_ outlook com

home sandesh vihar pitampura new delhi 110034

date of birth 7th april 1998

language english hindi french


work experience

what after college 4 month

delhi india

creating content to teach core java and python with data structure and
algorithm and giving online class to student


summer research fellow at university of auckland 2 month

auckland new zealand

worked on geometry of mobius transformation differential geometry under dr
pedram hekmati at the department of

mathematics university of auckland
```

software developer at cern 14 month

cern geneva switzerland

worked in the core platform team of the fap bc group part of an agile team of developer that maintains and add core

functionality to application used internally at cern by hr financial administrative and other department

including scientific

worked on legacy application that comprise of single and some time multiple framework such a java spring boot

hibernate and java ee also worked with google polymer 1 0 and jsp on the client side

maintained cern s electronic document handing system application with 1m loc that comprising of multiple framework

and created 20 year ago worked on feature request support request and incident and also release cycle

teaching assistant 4 month

coding ninja delhi

served a the teaching assistant to nucleus java with d batch under mr ankur kumar worked on creating course

content and quiz for online platform of coding ninja for java helped student in core data structure and

algorithm concept in java

education

delhi technological university 2016 2021

bachelor of technology mathematics and computing

cgpa 9 2

the heritage school rohini 2004 2016

physic chemistry math computer science with english

senior secondary 94 8

secondary 9 8 cgpa

technical skill

java algorithm and data structure

mean stack web development

python machine learning

matlab octave

mysql postgressql mongodb

other skill

m office adobe photoshop latex mitex

university course

applied mathematics i ii iii

linear algebra probability statistic stochastic process discrete math

computer organization architecture data structure algorithm design and analysis dbms o

computer vision nlp

important link

http www linkedin com in anishsachdeva1998

http github com anishlearnstocode

http www hackerrank com anishviewer

Analytics

We create the analytics.py file that will load the stemmed resume and lemmatized resume from the resume_stemmed.p and resume_lemmatized.p files respectively. It will then uses the word_tokenize method from the nltk.tokenize module to get the word tokens from both the resume outputs and will basic analysis tasks on both these token lists.

analytics.py *[see on GitHub]*

```
import pickle
import nltk
import pprint
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
nltk.download('wordnet')



def get_pos_frequency(tokens: list) -> Counter:
  synsets = [wordnet.synsets(token) for token in tokens]
  pos_tags = []
  for synset in synsets:
    if isinstance(synset, list) and len(synset) > 0:
      pos_tags.append(synset[0].pos())
  return Counter(pos_tags)



resume_file = open('../assets/resume.txt', 'r')
resume = resume_file.read().lower()
resume_file.close()
resume_stemmed = pickle.load(open('../assets/resume_stemmed.p', 'rb'))
resume_lemmatized = pickle.load(open('../assets/resume_lemmatized.p', 'rb'))
```

```
# extracting tokens from both the stemmed and lemmatized outputs
resume_tokens = word_tokenize(resume)
stemmed_resume_tokens = word_tokenize(resume_stemmed)
lemmatized_resume_tokens = word_tokenize(resume_lemmatized)

# Comparing the number of tokens in original, stemmed and lemmatized outputs
print('No. of tokens in Resume:', len(resume_tokens))
print('No. of tokens in Stemmed Resume:', len(stemmed_resume_tokens))
print('No. of tokens in Lemmatized Resume:', len(lemmatized_resume_tokens))

# comparing no. of words and word frequencies in both stemmed and lemmatized outputs
stemmed_resume_frequencies = Counter(stemmed_resume_tokens)
lemmatized_resume_frequencies = Counter(lemmatized_resume_tokens)
print('\nNo. of unique tokens/words in the stemmed output:',
len(stemmed_resume_frequencies))
print('No. of unique tokens/words in the lemmatized output:',
len(lemmatized_resume_frequencies))

# seeing the top 30 most common words in the stemmed and lemmatized outputs
print('\nTop 30 most common words/tokens in the stemmed output:\n',
stemmed_resume_frequencies.most_common(30))
print('Top 30 most common words/tokens in the lemmatized output:\n',
lemmatized_resume_frequencies.most_common(30))

# Analyzing of frequency of POS tags in original, stemmed and Lemmatized resume
resume_pos_frequency = get_pos_frequency(resume_tokens)
stemmed_resume_pos_frequency = get_pos_frequency(stemmed_resume_tokens)
lemmatized_resume_pos_frequency = get_pos_frequency(lemmatized_resume_tokens)

print('\nResume POS Tags Frequency:', resume_pos_frequency)
print('Stemmed Resume POS Tags Frequency:', stemmed_resume_pos_frequency)
print('Lemmatized Resume POS Tags Frequency:', lemmatized_resume_pos_frequency)
```

```
# compiling pos tags for words that have been stemmed and lemmatized differently
printer = pprint.PrettyPrinter(width=50)
diff = []

for stemmed, lemmatized in zip(stemmed_resume_tokens, lemmatized_resume_tokens):
  if not stemmed == lemmatized:
    stemmed_synsets = wordnet.synsets(stemmed)
    lemmatized_synsets = wordnet.synsets(lemmatized)
    stemmed_synset = stemmed_synsets[0].pos() if isinstance(stemmed_synsets, list) and len(stemmed_synsets) > 0 else ''
    lemmatized_synset = lemmatized_synsets[0].pos() if isinstance(lemmatized_synsets, list) and len(lemmatized_synsets) > 0 else ''
    diff.append({stemmed: stemmed_synset, lemmatized: lemmatized_synset})
print('POS Tags for Words with different stemmed and Lemmatized forms:')
printer.pprint(diff)
```

Output

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\anish\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
No. of tokens in Resume: 424
No. of tokens in Stemmed Resume: 363
No. of tokens in Lemmatized Resume: 363

No. of unique tokens/words in the stemmed output: 219
No. of unique tokens/words in the lemmatized output: 228

Top 30 most common words/tokens in the stemmed output:
 [('and', 16), ('of', 12), ('work', 7), ('java', 7), ('the', 6),
('with', 5), ('on', 5), ('develop', 4), ('com', 4), ('delhi', 4),
('month', 4), ('to', 4), ('core', 4), ('data', 4), ('structur', 4),
```

```
('algorithm', 4), ('at', 4), ('univers', 4), ('cern', 4), ('in', 4),
('comput', 4), ('code', 3), ('creat', 3), ('teach', 3), ('auckland',
3), ('mathemat', 3), ('that', 3), ('applic', 3), ('http', 3),
('softwar', 2)]
```

Top 30 most common words/tokens in the lemmatized output:

```
 [('and', 16), ('of', 12), ('java', 7), ('worked', 6), ('the', 6),
('with', 5), ('on', 5), ('com', 4), ('delhi', 4), ('month', 4),
('to', 4), ('core', 4), ('data', 4), ('structure', 4), ('algorithm',
4), ('at', 4), ('university', 4), ('cern', 4), ('in', 4),
('developer', 3), ('auckland', 3), ('mathematics', 3), ('that', 3),
('application', 3), ('computer', 3), ('http', 3), ('software', 2),
('new', 2), ('english', 2), ('4', 2)]
```

```
Resume POS Tags Frequency: Counter({'n': 203, 'v': 20, 'a': 18, 's':
10, 'r': 4})

Stemmed Resume POS Tags Frequency: Counter({'n': 161, 'a': 11, 'v':
10, 's': 7, 'r': 4})

Lemmatized Resume POS Tags Frequency: Counter({'n': 215, 'v': 20,
'a': 18, 's': 11, 'r': 5})
```

POS Tags for Words with different stemmed and Lemmatized forms:

```
[{'softwar': '', 'software': 'n'},
 {'develop': 'v', 'developer': 'n'},
 {'languag': '', 'language': 'n'},
 {'experi': '', 'experience': 'n'},
 {'colleg': '', 'college': 'n'},
 {'creat': '', 'creating': 'v'},
 {'structur': '', 'structure': 'n'},
 {'give': 'n', 'giving': 'n'},
 {'onlin': '', 'online': 'a'},
 {'univers': '', 'university': 'n'},
 {'work': 'n', 'worked': 'v'},
 {'geometri': '', 'geometry': 'n'},
 {'mobiu': '', 'mobius': 'n'},
```

```
{'transform': 'v', 'transformation': 'n'},
{'differenti': '', 'differential': 'n'},
{'geometri': '', 'geometry': 'n'},
{'depart': 'v', 'department': 'n'},
{'mathemat': '', 'mathematics': 'n'},
{'univers': '', 'university': 'n'},
{'softwar': '', 'software': 'n'},
{'develop': 'v', 'developer': 'n'},
{'work': 'n', 'worked': 'v'},
{'agil': '', 'agile': 's'},
{'develop': 'v', 'developer': 'n'},
{'maintain': 'v', 'maintains': 'v'},
{'function': 'n', 'functionality': 'n'},
{'applic': '', 'application': 'n'},
{'us': 'n', 'used': 'v'},
{'intern': 'n', 'internally': 'r'},
{'financi': '', 'financial': 'a'},
{'administr': '', 'administrative': 'a'},
{'depart': 'v', 'department': 'n'},
{'includ': '', 'including': 'v'},
{'scientif': '', 'scientific': 'a'},
{'work': 'n', 'worked': 'v'},
{'legaci': '', 'legacy': 'n'},
{'applic': '', 'application': 'n'},
{'compris': '', 'comprise': 'v'},
{'singl': '', 'single': 'n'},
{'multipl': '', 'multiple': 'n'},
{'hibern': '', 'hibernate': 'v'},
{'work': 'n', 'worked': 'v'},
```

```
{'googl': '', 'google': 'n'},
{'polym': '', 'polymer': 'n'},
{'maintain': 'v', 'maintained': 'v'},
{'electron': 'n', 'electronic': 'a'},
{'hand': 'n', 'handing': 'v'},
{'applic': '', 'application': 'n'},
{'compris': '', 'comprising': 'v'},
{'multipl': '', 'multiple': 'n'},
{'creat': '', 'created': 'v'},
{'work': 'n', 'worked': 'v'},
{'featur': '', 'feature': 'n'},
{'incid': '', 'incident': 'n'},
{'releas': '', 'release': 'n'},
{'cycl': '', 'cycle': 'n'},
{'teach': 'n', 'teaching': 'n'},
{'assist': 'n', 'assistant': 'n'},
{'code': 'n', 'coding': 'n'},
{'serv': '', 'served': 'v'},
{'teach': 'n', 'teaching': 'n'},
{'assist': 'n', 'assistant': 'n'},
{'nucleu': '', 'nucleus': 'n'},
{'work': 'n', 'worked': 'v'},
{'creat': '', 'creating': 'v'},
{'cours': '', 'course': 'n'},
{'quiz': 'n', 'quizz': ''},
{'onlin': '', 'online': 'a'},
{'code': 'n', 'coding': 'n'},
{'help': 'n', 'helped': 'v'},
{'structur': '', 'structure': 'n'},
```

```
{'educ': '', 'education': 'n'},
{'technolog': '', 'technological': 's'},
{'univers': '', 'university': 'n'},
{'technolog': '', 'technology': 'n'},
{'mathemat': '', 'mathematics': 'n'},
{'comput': '', 'computing': 'n'},
{'heritag': '', 'heritage': 'n'},
{'chemistri': '', 'chemistry': 'n'},
{'comput': '', 'computer': 'n'},
{'scienc': '', 'science': 'n'},
{'secondari': '', 'secondary': 'n'},
{'secondari': '', 'secondary': 'n'},
{'technic': '', 'technical': 'n'},
{'structur': '', 'structure': 'n'},
{'develop': 'v', 'development': 'n'},
{'machin': '', 'machine': 'n'},
{'learn': 'v', 'learning': 'n'},
{'octav': '', 'octave': 'n'},
{'offic': '', 'office': 'n'},
{'adob': '', 'adobe': 'n'},
{'univers': '', 'university': 'n'},
{'cours': '', 'course': 'n'},
{'appli': '', 'applied': 'v'},
{'mathemat': '', 'mathematics': 'n'},
{'probability': 'n', 'probabl': ''},
{'statist': '', 'statistic': 'n'},
{'stochast': '', 'stochastic': 's'},
{'discret': '', 'discrete': 's'},
{'comput': '', 'computer': 'n'},
```

```
 {'organ': 'n', 'organization': 'n'},
 {'architectur': '', 'architecture': 'n'},
 {'structur': '', 'structure': 'n'},
 {'analysi': '', 'analysis': 'n'},
 {'dbm': '', 'dbms': 'n'},
 {'comput': '', 'computer': 'n'},
 {'import': 'n', 'important': 'a'},
 {'anishlearnstocod': '',
  'anishlearnstocode': ''},
 {'anishview': '', 'anishviewer': ''}]

Process finished with exit code 0
```

Discussion

From the analytics output produced above we can make some important inferences:

Both stemming and Lemmatization serve a very similar purpose in the NLP preprocessing pipeline and are methods used to reduce a word to it's toot form, they also have many differences.

Stemming reduces a word by removing it's inflections and the morphologically reduced form that it receives may or may not be an english language word. Stemming uses a heuristic approach to reducing inflections wherein the steps followed are not aware of the meaning of the word or even the context of the word and simply follow basic deterministic rules based on the characters in the word to add and remove character strings in the word.

This isn't how lemmatization functions. The Porter Stemmer Algorithm was introduced in the 1980's whereas Lemmatization is a much more modern algorithm that isn't a simple heuristic based algorithm, but it is aware of the word meaning along with synonyms of the word.

It also tags the word with the correct part of speech (POS) tag, or multiple tags and uses a pre-built dictionary to define the meaning of each and every tag with different contexts (Contexts being Noun, Verb, Adverb etc.) Lemmatization requires this pre-built dictionary to function and in the **`nltk`** package that we were using, internally **`nltk`** uses the **`wordnet`** corpus which contains words, their definitions for all different contexts and also synonyms like a thesaurus.

Lemmatization using the wordnet corpus can reduce words to words that do not have the same character structure e.g. it can reduce *better* → *good* if we tell it that *better* here is a **verb**.

Stemming reduces words having the same character roots (may not have same meanings) to the same roots and this is then helpful in **IR** (Information Retrieval) Applications as the person/user can search for small strings like *uni* or *univer* and these strings will automatically match to *university* , *universities* etc.

So, stemming makes a lot of sense in Information Retrieval Applications. In advanced information retrieval applications where the user can not only enter a stemmed form of what she is searching, but can also enter the context of what she wishes to search such as *better food than x* (where x is a restaurant). Our IR application should be able to understand that *better* here refers to *good* food or better ratings than the ratings for a restaurant.

Or we may have a chatbot application which communicates with the user and the chatbot application needs to understand the intent of the user so that the chatbot can answer queries that the user puts forth. For answering queries or understanding speech and

translating to text, or calculating the probability of a given word we require a model that can understand context and not just the root of a word.

In such applications we use lemmatization along with POS (Part of Speech) tagging. Even in machine translations wherein we need to compute the probabilities of the translated text, we need a lemmatizer along with a POS tagger to compute structure and probabilities.

Hence, both the stemmer and the lemmatizer cater to very different needs. In our application we decide whether to use a stemmer or lemmatizer based on what our application must do. In a system with multiple resumes, the most common thing an employer might want to do is search the corpora with specific skills such as *management*, *java*, *machine learning* etc. and then receive resumes with a match for this string.

We can also sort the resumes based on frequency and count of matches, where the resume with a higher number of matches might bubble up to the top. Hence, in our application which is centered more around Information Extraction/Information Retrieval than context understanding Stemming makes more sense.

This may seem counterintuitive as we have seen in the analytics above that lemmatization preserves the POS tags and context structure, but preserving POS tags and context structure will not improve an IR system. Also running a lemmatizer is more compute heavy as it has a higher time complexity.

Bibliography

1. Speech & Language Processing ~Jurafsky
2. nltk
3. pickle
4. Porter Stemmer Algorithm
5. Porter Stemmer Implementation ~anishLearnsToCode
6. NLTK WordNetInterface
7. NLTK Stemming Submodule