Python | Lemmatization with NLTK

Lemmatization is a fundamental **text pre-processing technique** widely applied in **natural language processing (NLP)** and machine learning. Serving a purpose akin to stemming, lemmatization seeks to distill words to their foundational forms. In this linguistic refinement, the resultant base word is referred to as a "lemma." The article aims to explore the use of lemmatization and demonstrates how to perform **lemmatization with NLTK**.

Table of Content

Lemmatization

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meanings to one word.
Text preprocessing includes both Stemming as well as lemmatization. Many times, people find these two terms confusing. Some treat these two as the same. Lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

> Examples of lemmatization:
>
> -> rocks : rock
>
> -> corpora : corpus
>
> -> better : good

One major difference with stemming is that lemmatize takes a part of speech parameter, "pos" If not supplied, the default is "noun."

Lemmatization Techniques

Lemmatization techniques in natural language processing (NLP) involve methods to identify and transform words into their base or root forms, known as lemmas. These approaches contribute to text

normalization, facilitating more accurate language analysis and processing in various NLP applications. Three types of lemmatization techniques are:

1. Rule Based Lemmatization

Rule-based lemmatization involves the application of predefined rules to derive the base or root form of a word. Unlike machine learning-based approaches, which learn from data, rule-based lemmatization relies on linguistic rules and patterns.

Here's a simplified example of rule-based lemmatization for English verbs:

Rule: For regular verbs ending in "-ed," remove the "-ed" suffix.

Example:

- Word: "walked"
- Rule Application: Remove "-ed"
- Result: "walk

This approach extends to other verb conjugations, providing a systematic way to obtain lemmas for regular verbs. While rule-based lemmatization may not cover all linguistic nuances, it serves as a transparent and interpretable method for deriving base forms in many cases.

2. Dictionary-Based Lemmatization

Dictionary-based lemmatization relies on predefined dictionaries or lookup tables to map words to their corresponding base forms or lemmas. Each word is matched against the dictionary entries to find its lemma. This method is effective for languages with well-defined rules.

Suppose we have a dictionary with lemmatized forms for some words:

- 'running' → 'run'
- 'better' → 'good'
- 'went' → 'go'

When we apply dictionary-based lemmatization to a text like "I was running to become a better athlete, and then I went home," the resulting lemmatized form would be: "I was run to become a good athlete, and then I go home."

3. Machine Learning-Based Lemmatization

Machine learning-based lemmatization leverages computational models to automatically learn the relationships between words and their base forms. Unlike rule-based or dictionary-based approaches,

machine learning models, such as neural networks or statistical models, are trained on large text datasets to generalize patterns in language.

Example:

Consider a machine learning-based lemmatizer trained on diverse texts. When encountering the word 'went,' the model, having learned patterns, predicts the base form as 'go.' Similarly, for 'happier,' the model deduces 'happy' as the lemma. The advantage lies in the model's ability to adapt to varied linguistic nuances and handle irregularities, making it robust for lemmatizing diverse vocabularies.

Implementation of Lemmatization

NLTK

Below is the implementation of lemmatization words using NLTK

Python3

```
from  nltk.stem  import  WordNetLemmatizer

lemmatizer  =  WordNetLemmatizer()

print``(``"rocks :"``, lemmatizer.lemmatize(``"rocks"``))

print``(``"corpora :"``, lemmatizer.lemmatize(``"corpora"``))

print``(``"better :"``, lemmatizer.lemmatize(``"better"``, pos``=``"a"``))
```

Output:

```
rocks : rock
corpora : corpus
better : good
```

NLTK (Natural Language Toolkit) is a Python library used for natural language processing. One of its modules is the WordNet Lemmatizer, which can be used to perform lemmatization on words.

Lemmatization is the process of reducing a word to its base or dictionary form, known as the lemma. For example, the lemma of the word "cats" is "cat", and the lemma of "running" is "run".

Spacy

Python3

```
import  spacy

nlp  =  spacy.load(``'en_core_web_sm'``)

text  =  "The quick brown foxes are jumping over the lazy dogs."

doc  =  nlp(text)

lemmatized_tokens  =  [token.lemma_  for  token  in  doc]

lemmatized_text  =  ' '``.join(lemmatized_tokens)

print``(``"Original Text:"``, text)

print``(``"Lemmatized Text:"``, lemmatized_text)
```

Output:

```
Original Text: The quick brown foxes are jumping over the lazy dogs.
Lemmatized Text: the quick brown fox be jump over the lazy dog .
```

Advantages of Lemmatization with NLTK

1. **Improves text analysis accuracy:** Lemmatization helps in improving the accuracy of text analysis by reducing words to their base or dictionary form. This makes it easier to identify and analyze words that have similar meanings.
2. **Reduces data size:** Since lemmatization reduces words to their base form, it helps in reducing the data size of the text, which makes it easier to handle large datasets.
3. **Better search results:** Lemmatization helps in retrieving better search results since it reduces different forms of a word to a common base form, making it easier to match different forms of a word in the text.

Disadvantages of Lemmatization with NLTK

1. **Time-consuming:** Lemmatization can be time-consuming since it involves parsing the text and performing a lookup in a dictionary or a database of word forms.
2. **Not suitable for real-time applications:** Since lemmatization is time-consuming, it may not be suitable for real-time applications that require quick response times.

3. **May lead to ambiguity:** Lemmatization may lead to ambiguity, as a single word may have multiple meanings depending on the context in which it is used. In such cases, the lemmatizer may not be able to determine the correct meaning of the word.

Also Check:

- Removing stop words with NLTK in Python
- Python | PoS Tagging and Lemmatization using spaCy
- Python | Named Entity Recognition (NER) using spaCy

Frequently Asked Questions (FAQs)

1. What is lemmatization?

> Lemmatization is the process of reducing a word to its base or root form, typically by removing inflections or variations. It aims to group together different forms of a word to analyze them as a single entity.

2. How is lemmatization different from stemming?

> While both lemmatization and stemming involve reducing words to their base forms, lemmatization considers the context and morphological analysis to return a valid word, whereas stemming applies simpler rules to chop off prefixes or suffixes, often resulting in non-dictionary words.

3. Why is lemmatization important in natural language processing (NLP)?

> Lemmatization is crucial in NLP for tasks such as text analysis, sentiment analysis, and information retrieval. It helps in standardizing words, reducing dimensionality, and improving the accuracy of language processing models.

4. How does lemmatization handle different parts of speech?

> Lemmatization takes into account the grammatical category of a word (noun, verb, adjective, etc.) and provides the base form accordingly. For example, the lemma of “running” as a verb is “run,” while as a noun, it remains “running.”

5. What are some common lemmatization tools or libraries in Python?

> Popular libraries for lemmatization in Python include NLTK (Natural Language Toolkit), spaCy, and the TextBlob library. Each library may have its own set of rules and algorithms for lemmatization.