International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)

SURVEY ON COMPREHENSIVE TECHNIQUES OF TEXT DATA AUGMENTATION

Ritu Khare, Anshuman Mahapatra
Data and AI
Advanced Technology Centers in India, Accenture
Email: ritu.khare@accenture.com
Email: anshuman.a.mahapatra@accenture.com

Abstract: **In the world of Data Science and machine learning everything starts with data and how well one can use it to get the desired output. But what can be done when enough data is not present. Obviously, overall result will get impacted because of scarcity of data. Hence, to resolve this problem, data augmentation comes into picture which helps to increase dataset size by augmenting it. Data augmentation is an assortment of techniques that facilitates to automatically generate high quality data with the help of existing data. In the field of Natural Language Processing (NLP) is difficult to augment the text due to huge complexity of language. The process of augmenting text data is more challenging and not as straightforward as one might expect. In this study, the way of doing NLP data augmentation and libraries available for it is explored. Data augmentation techniques, comparison and features of the python libraries which can be used for data augmentation is discussed. It will help researchers and data scientist to decide which library to use for their job.**

Keywords: **Natural Language Processing (NLP), Text Data Augmentation, Data Augmentation Libraries, Training Data Augmentation**

I. INTRODUCTION

Data is essential for machine learning models to perform optimally, but annotating a large amount of data can be a costly endeavor. It is important to not only focus on having a lot of data but having the right data and the right techniques to ensure the best performance possible. Therefore, proper data augmentation is useful to boost up model performance [1]. Augmentation is an immensely popular technique in the computer vision field. It involves the alteration of an image to create a new one. This can be done in many ways, such as flipping, adding noise, adjusting the brightness and contrast, and cropping. Augmentation can be used to help reduce over fitting, making models more robust. It is established that augmentation is one of the solutions to the success of computer vision models.

In natural language processing (NLP), text augmentation is a difficult task to undertake due to the complexity of the language. While some words can be replaced with others, such as a, an, the, this is often not sufficient to adequately enhance a piece of text. Also, not every word has a synonym [2]. The complexities of language are vast, making it difficult to know which words to choose when attempting to alter the meaning of a sentence. Even the slightest variation in word choice can have a remarkable effect on the context of a statement. Additionally, the context of the sentence must be considered to ensure that the meaning is preserved. It is for these reasons that text augmentation is especially difficult in the field of NLP. Alternatively, augmented images can be generated relatively easily using computer vision. Even when noise is introduced or portions of the image are cropped out, the model can still classify the image correctly. Data Augmentation (DA) Technique is an effective way of increasing the size and quality of a dataset without having to manually collect more data. The data needs to be altered to preserve the class categories for better performance in the classification task [3].

Data augmentation is a powerful tool for improving the performance of machine learning models. By increasing the size of the training data, it can help to create better models that are more robust and perform better on unseen data. However, when augmenting data, it is important to ensure that the distribution of the augmented data generated is neither too similar nor too different from the original as this can lead to over fitting or inferior performance. Effective data augmentation approaches should aim to achieve a balance between the two, as this can help to ensure that the model has seen a variety of different data points.

If you have a tiny dataset, data augmentation can be a particularly effective way of increasing the size of the training set. This can help to improve the model's performance and help to reduce the risk of over fitting. Additionally, data augmentation can help to ensure that the model has seen a variety of different data points, which can help to improve the model's generalization performance. By making sure that the data is neither too similar nor too different from the original, it can help to ensure that the model is able to make accurate predictions on unseen data.

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)

It may be a promising idea to apply data augmentation approaches for small dataset with aim to reduce over fitting in machine learning models.

II. TEXT DATA AUGMENTATION:

Data Augmentation can create more data for machine learning project. In the field of text data augmentation there are many methods available on character, word and sentence level to generate quality data without losing the meaning and context of original text. It gives a promising result in terms of quality as well as quantity of data if done wisely based on requirement and available data. There are various techniques to do so. Let's discuss them in detail.

1.1 Text Data Augmentation Methods:

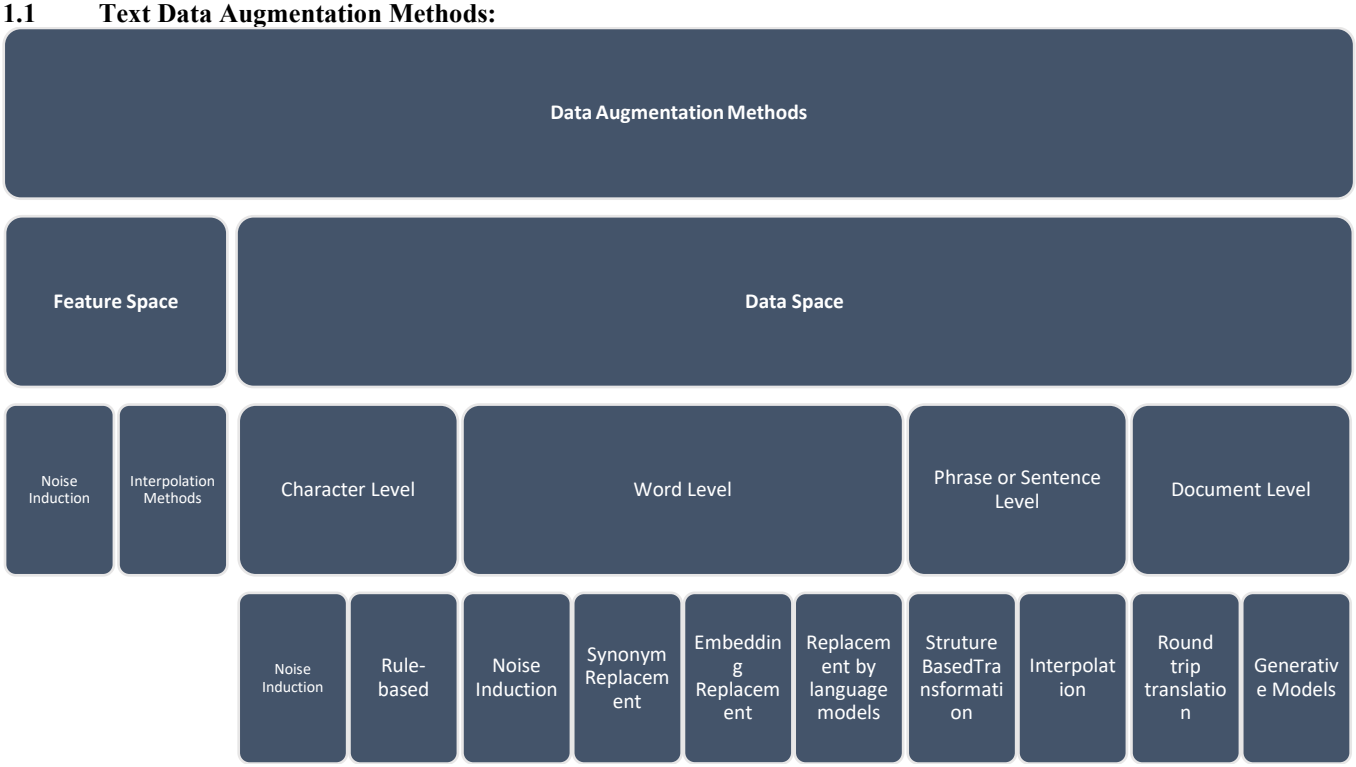

Fig 1. Data Augmentation Methods in Feature and Data Space

1.1.1 Data Augmentation in Feature Space

In the feature space, data augmentation is the process of transforming the input data into latent vector representations (feature form) of the inputs. The feature space has two types of data augmentation: noise induction and interpolation [4].

Noise Induction: In the data space, noise can be introduced in many ways, and the feature space is no different. It is possible to introduce random multiplicative and additive noise to feature representations, which can give a variety of results. These noise variants can be used to measure the robustness of a model, as well as to improve performance on certain tasks. Additionally, artificially introducing noise can also help to reduce the risk of over fitting, as it can help to better generalize the learned model.

Interpolation Methods: In this method a new sentence can be created by taking the hidden states of two sentences and combining them, resulting in a sentence that retains the meaning of both original sentences.

1.1.2 Data Augmentation in Data Space

Data augmentation in the data space allows for the transformation of raw data into a more readable and usable form. There are four types of data augmentation techniques available in the data space: character level, word level, phrase and sentence level, and document level [5].

Character Level

In the case of character-level augmentation creating new training samples from existing ones by changing single characters. This is done by randomly selecting characters from the existing sample and replacing them with new ones. This technique can be especially useful when the dataset is small or limited. By randomly changing characters, a variety

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)

of new samples can be created that can help model to better generalize and learn from different types of data.

- **Noise Induction:** It involves random character deletion, swap, and insertion.
- **Rule-based Transformations:** Regular expressions can be used to insert spelling mistakes, alter data, abbreviate entity names, and insert spelling mistakes.

Word Level

In this type of data augmentation, single words are changed entirely to create new training samples from existing ones [6].

- **Noise Induction:**"Unigram noising" involves replacing words in input data with another word with a certain probability. Using "blank noise", words are replaced with "_", while random word substitutions and deletions are also methods of inducing noise.
- **Synonym Replacement:** In this very popular form of data augmentation, certain words are replaced with synonyms in text instances. Synonym replacement is usually performed using knowledge bases like WordNet.
- **Embedding Replacement:** Comparable to synonym substitution, embedding replacement methods seek to find words that fit into the textual context without changing the fundamental meaning of the text. This is done by transforming words into a latent representation space, where words with similar contexts are closer together. Then, a word is replaced by the one which is closest to it in the latent representation space. This ensures that the contextual meaning of the text remains the same while still providing the desired level of variation.
- **Replacement by Language Models:** Language models represent language by predicting subsequent or missing words based on the previous or surrounding context. This allows them to be used to filter unfitting words and produce more accurate results than those obtained via global context-based embedding replacements. By focusing on the local context, language models can improve the accuracy of language understanding and filtering of bad words produced with the embedding replacement techniques.

Phrase and Sentence Level

This type of data augmentation consists of changing sentence structures to create new training samples [15].

- **Structure-based Transformation:** The structure-based approach to data augmentation involves the use of certain features or components of the structures to generate modified texts. Grammatical formalities, such as dependency and constituent grammars or POS-tags, can be used as the basis for such structures. One method of data augmentation is cropping sentences and putting the focus on the subjects and objects. The 'rotation' technique is a flexible method that involves moving fragments of words around. This approach is useful as it can generate new sentences from existing ones, allowing for the creation of a larger data set. It also allows for different contexts and relationships to be explored.
- **Interpolation:** This method of natural language processing works by substituting substructures of the training examples that have the same tagged label. For instance, an example sentence substructure of "a [DT] chocolate [NN]" (where [DT] and [NN] are Determiner and Singular Noun respectively) can be replaced with a new sentence substructure of "a [DT] cat [NN]" from another instance, resulting in an interpolation. This substitution process allows for additional insight as to the relationship between different words with the same tag. By replacing words with similar tags, it can be determined how two words might be related to one another. Additionally, this process helps to create a larger, more diverse dataset that can be used to train a machine learning model.

Document Level

The purpose of this type of data augmentation is to create new training samples from existing ones by modifying entire sentences in the documents.

- **Round-trip Translation:** Round-trip translation is an approach to creating paraphrases with the assistance of translation models. In this approach, a word, phrase, sentence, or document is translated from its source language into another language (forward translation) and then reversed back into the source language (back-translation). This method allows to produce multiple paraphrases that adhere to the same meaning of the original source, while also providing a unique spin on the wording. As such, round-trip translation is a valuable tool for creating more accurate translations and paraphrases.
- **Generative Methods:** A significant increase in language generation capabilities allowed the current models to create very diverse texts that can incorporate new information, which in turn led to a significant increase in text generation capabilities. A document-level data augmentation method uses training language models (VAEs, RNNs, Transformers) to produce documents that look like the training data.[16]

III. TEXT DATA AUGMENTATION LIBRARIES

There are several types of data augmentation libraries present based on the feature types and tasks user want to perform. We'll look at some of them:

a. Text Attack

b. NLP Aug
c. Google Trans
d. Text Augment
e. Aug Ly
f. Parrot Paraphraser
g. Pegasus Paraphraser

a. Text Attack

It is a Python library. It is used for combative attacks, combative training, and data augmentation in NLP. Text attack has a text attack. Augmenter class which provides six different approaches for data augmentation[7].

Text attack installation

```
!pip install text attack
```

i. **Word Net Augmenter**

The WordNet augmenter identifies synonyms from WordNet and replaces them with the appropriate words.

Implementation:

```
From text attack.augmentation import WordNetAugmenter
orignal_text = "Early to bed and early to rise makes a man healthy, wealthy and wise"
text_attack_wordnet = WordNetAugmenter()
augmented_text_attack_wordnet.augment(orignal_text)
print(augumented_text)
```

OUTPUT

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: ['Early to bed and early to rise makes a man healthful, wealthy and wise']

ii. **Embedding Augmenter**

This augmenter replaces words with neighbors in the counter-fitted embedding space to ensure their consine similarity is at least 0.8

Implementation:

```
from textattack.augmentation import EmbeddingAugmenter
text_attack_embed = EmbeddingAugmenter()
augmented_text=text_attack_embed.augment(orignal_text)
print(augumented_text)
```

OUTPUT

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: ['Early to bed and early to wake up makes a man healthy, wealthy and wise']

iii. **Easy Data Augmenter**

EDA augments text by replacing words, adding words, and removing words.

Implementation:

```
from textattack.augmentation import EasyDataAugmenter
text_attack_eda = EasyDataAugmenter()
text_attack_eda.augment(text)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: ['Early to bed early to rise makes a man healthy, wealthy and wise',
'Early to bed and early to rise makes a man healthy, prosperous and wise',
'Early to bed and early to rise makes a man healthy, wealthy and intelligent',
'Early to bed and early to rise makes a gentle man strong, wealthy and intelligent']

iv. **Char Swap Augmenter**

As parts of the augmentation process, it replaces, deletes, inserts, and swaps adjacent characters as needed.

Implementation:

```
from textattack.augmentation import CharSwapAugmenter
text_attack_charswap = CharSwapAugmenter()
text_attack_charswap.augment(text)
```

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to bed and early to rise amkes a man healthy, wealthy and wise'

v. **Check List Augmenter**

Names, locations, and numbers are compressed/extended and substituted in this augmenter.

Implementation:

```
from textattack.augmentation import Check List Augmenter
text_attack_checklist = Check List Augmenter()
text_attack_checklist.augment(text)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'

Augmented Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'

vi. CLARE Augmenter

Using a pre-trained masked language model, it augments text by replacing, inserting, and merging.

Implementation:

```
from textattack.augmentation import CLAREAugmenter
text_attack_clare = CLAREAugmenter()
print(text_attack_clare.augment(text))
```

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)

Augmented Text: 'Early to bed and early to rise makes a man extremely healthy, wealthy and wise'

b. **NLPAug**

It is a python-based augmentation library for machine learning experiments based on text. NLP Aug is a tool that can assist in this process by enhancing natural language processing (NLP) for machine learning applications. Besides generating textual data for training and testing models, it can generate adversarial examples that can be used to identify and prevent adversarial attacks. NLP Aug is a powerful tool for improving deep learning model performance and ensuring the security of machine learning applications. Let's look at how to use this library to enhance data.[8]

NLP Aug installation

!pip install nlpaug

NLP Aug provides basically three distinct types of augmentation based on character, word and sentence augmentation. Below is a list of available augmenters mapped with the type of augmentation:

Table 1.NLPAug Augmenters and its description

NLP Aug Augmenters	**Type of Augmentation**
Antonym Aug	Substitute opposite meaning word according to WordNet antonym
Random Aug	Apply insert, substitute, swap, delete augmentation randomly
Character Keyboard Aug	It substitutes by Simulating keyboard distance error
Ocr Aug	It substitutes by OCR engine error
Word Embs Aug	It substitutes, insert using word2vec, GloVe, or fast text embeddings to apply augmentation

i. Antonym Aug:

Implementation:

```
import nlpaug.augmenter.word as nawordaug = naword.Ant
onymAug(name='Antonym_Aug', aug_min=1, aug_max=10
, aug_p=0.3, lang='eng', stopwords=None, tokenizer=None,
revee_=None, stopwords_regex=None, verbose=0)
test_sentence_aug = aug.augment("too dark")
print(test_sentence_aug)
```

Original Text:"too dark"
Augmented Text:['too light']

ii. Random Aug:

Implentation:

```
import nlpaug.augmenter.word as naword
Text = 'Early to bed and early to rise makes a man healthy,
wealthy and wise'
aug = naword.RandomWordAug(action='delete', name='Ran
domWord_Aug', aug_min=1, aug_max=10, aug_p=0.3, stop
words=None, target_words=None, tokenizer=None, reverse
_tokenizer=None, stopwords_regex=None, verbose=0)
output = aug.augment (Text)
print(output)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text:['To and to rise makes a, wealthy and wise']

iii. Keyboard Aug:

Implementation:

```
import nlpaug.augmenter.char as na
Text='Early to bed and early to rise makes a man healthy,we
althy and wise'
aug = na.KeyboardAug(name='Keyboard_Aug', aug_char_
min=1, aug_char_max=10, aug_char_p=0.3, aug_word_p=0
.3, aug_word_min=1, aug_word_max=10, stopwords=None,
 tokenizer=None, reverse_tokenizer=None, include_special_
char=True, include_numeric=True, include_upper_case=Tru
e, lang='en', verbose=0, stopwords_regex=None, model_pat
h=None, min_char=4)
output = aug.augment(Text)
print(output)
```

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text:['EaGlJ to bed and eq$ly to ejsemsJes a man healthy, wealthy and SKse']

iv. Ocr Aug:

Implementation:

```
import nlpaug. augmenter. char as na
Text = 'Early to bed and early to rise makes a man healthy,
wealthy and wise'
aug = na.OcrAug(name='OCR_Aug', aug_char_min=1, aug
_char_max=10, aug_char_p=0.3, aug_word_p=0.3, aug_wor
d_min=1,
            aug_word_max=10, stopwords=None, tokenizer=
None, reverse_tokenizer=None, verbose=0, stopwords_rege
x=None,
            min_char=1)
```

```
output = aug.augment(Text)
print(output)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: ['Early tu bed and eak1y to rise makes a man hea1thy, wealthy and wise']

v. Word Embs Aug
Implementation:

```
!pip install transformers
import nlpaug. augmenter. word as naword
TOPK=20 #default=100
ACT = 'insert' #"substitute"
Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'
aug_bert = naword.ContextualWordEmbsAug(model_path='distilbert-base-uncased',action=ACT, top_k=TOPK)
for i in range(7):
output = aug_bert.augment(Text)
print(output)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'

Augmented Text:
['early steps to bed early and early to rise sun makes a man very healthy, wealthy citizen and wise']
['early to bed and very early to early rise he makes up a man reasonably healthy, wealthy and wise']
['early transition to bed habits and sometime early to mid rise makes a man healthy, wealthy looking and wise']
['early to bed hours and relatively early to rise makes a rich man reasonably healthy, wealthy and relatively wise']
['happy early retirement to reach bed rest and early to rise makes a healthy man healthy, wealthy and wise']
['early exposure to bed and early to nightly rise gradually makes that a man grow healthy, wealthy and wise']
['mornings early to bed and early to dusk rise makes a man extremely healthy, economically wealthy and morally wise']

c. Google trans
Itis built on top of Google Translate API. This uses Google Translate Ajax API for language detection and translation. [9]

Installation

```
!pip install google trans
```

Usage
The key parameters to translate() method are:
src: source language. Optional parameter as googletrans will detect it.
dest: destination language. Mandatory parameter.
text: the text which has to be translated from source language to the destination language. Mandatory parameter.

Implementation:
Translation from English to Chinese

```
from googletrans import Translator
translator = Translator()
my_translation = translator.
translate("Early to bed and early to rise makes a man wealth y, healthy and wise.", src='en', dest='zh-CN')
print(my_translation.text)
```

OUTPUT:
早睡早起使人富有、健康和聪明。

Translation from Chinese back to English

```
From google trans import Translator
translator = Translator()
my_translation = translator.
translate("早睡早起使人富有、健康和聪明。", src='zh-CN', dest='en')
print(my_translation.text)
```

OUTPUT:
Early to bed and early to rise makes a man rich, healthy and wise.
As in the above implementation, the given text is first translated from English to Chinese and then translated back to English. During this back translation, there is a slight change in the sentence between the original text and the back-translated text, but the overall meaning of the sentence is still retained.
Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to bed and early to rise makes a man rich, healthy and wise.'

d. Text Augment
The Text Augment library provides text augmentation for natural language processing use in Python. It enables users to add more data quickly and easily to their existing datasets. Text Augment provides a wide range of features including text generation, text substitution, text insertion, and text deletion. Text Augment works with the help of libraries like NLTK, Gensim, and Text Blob and works well with them.[11]

Installation:

```
pip install num pynltkgensim text blobgoogletrans text augment
```

i. Synonym Replacement
Implementation:

```
from text augment import EDA
```

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)


```
eda_augment= EDA()
Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Text_augument_syn=eda_augment.synonym_replacement( Text)
print(Text_augument_syn)
```

OUTPUT
Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to bed and early to rise makes a man nutritious, wealthy and wise'

ii. Random Deletion
Implementation:

```
from text augment import EDA
eda_augment= EDA()

Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'

random_del_augment=eda_augment.random_deletion(Text, p=0.2)

print (random_del_augment)
```

OUTPUT
Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to bed early to rise makes a man healthy, wealthy'

iii. Random Swap
Implementation:

```
From text augment import EDA

eda_augment= EDA()

Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'

random_swap_augment=eda_augment.random_swap(Text)
print(random_swap_augment)
```

OUTPUT
Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to bed and early to rise makes a man wise, wealthy and healthy'

iv. Random Insertion
Implementation:

```
from text augment import EDA
eda_augment= EDA()
Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'

rnd_insert_augment=eda_augment.random_insertion(Text)
print(rnd_insert_augment)
```

OUTPUT
Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'Early to clean bed and early to rise makes a man healthy, wealthy and wise''

e. Aug Ly

AugLy package is released by Facebook to the public domain. It is a prevalent library that is divided into four sub-libraries, each offering to a specific data mode. These sub libraries are audio, images, videos, and texts giving users a range of choices for data management [10].

Installation:

```
pip install -U augly
```

i. Replace Similar Characters
Implementation:

```
importaugly.textastextaugs
Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'

augly_aug=textaugs.replace_similar_chars(Text)
print(augly_aug)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: '3arly to bed and early to 7ise makes a man healthy, wealthy and wise'

ii. **Insert Punctuations**
Implementation:

```
import augly. text as textaugs
Text = 'Early to bed and early to rise makes a man healthy, wealthy and wise'

augly_aug=textaugs.insert_punctuation_chars(Text)
print(augly_aug)
```

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text: 'E.a.r.l.y. .t.o. b.e.d.. a.n.d.. e.a.r.l.y. .t.o..r.i.s.e..m.a.k.e.s..a..m.a.n..h.e.a.l.t.h.y,..w.e.a.l.t.h.y..a.n.d..w.i.s.e.'

f. Parrot Paraphraser

Parrot is a paraphrase-based augmentation framework which is built to accelerate training NLU models. It is more than just a simple paraphrasing model, however; it is designed to provide an extra layer of depth to NLU training. Parrot's

paraphrasing tool helps to create a variety of variations on a given sentence, allowing for more robust and expansive training. [12]

Installation:
pip install
git+https://github.com/PrithivirajDamodaran/Parrot_Paraphraser.git

Implementation:
Parrot Paraphraser Implementation
```
import torch
from parrot import Parrot
PARROT_PRETRAINED_MODEL = "prithivida/parrot_paraphraser_on_T5"
parrot_model = Parrot(model_tag=PARROT_PRETRAINED_MODEL)
PHRASE = "Can you recommend some great restaurants in Mumbai?"
para_phrases = parrot_model.augment(input_phrase=PHRASE, use_gpu=False)
for para_phrase in para_phrases:
print(para_phrase)
```

Original Text:'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text:
('tell me the best place to eat in mumbai?', 41)
('list some good restaurants in mumbai?', 32)
('tell me the best restaurant in mumbai?', 32)
('can you suggest some of the best restaurants in mumbai?', 29)
('can you list the best restaurants in mumbai?', 27)
('recommend some good restaurant in mumbai?', 25)

g. Pegasus Paraphraser

PEGASUS is a standard Transformer encoder-decoder which uses GSG (Gated Self-attention Graph) for pre-training on large corpora of documents [13].The encoder is trained to create a vector representation of the input sentence, while the decoder is trained to generate a response sentence given the vector representation of the input sentence. Both encoder and decoder are trained in a self-supervised manner, with the goal of obtaining a model that can accurately reconstruct an input sentence [14].

Installation:
Pip install transformers

Pegasus Implementation:
```
import torch
from transformers import PegasusForConditionalGeneration, PegasusTokenizer
BEAM_NUM = 10
RETURN_SEQ_NUM = 10
PEGASUS_PRETRAINED_MODEL = 'tuner007/pegasus_paraphrase'
pegasus_tokenizer = PegasusTokenizer.from_pretrained(PEGASUS_PRETRAINED_MODEL)
torch_device = 'cuda' if torch.cuda.is_available() else 'cpu'
pegasus_model = PegasusForConditionalGeneration.from_pretrained(PEGASUS_PRETRAINED_MODEL).to(torch_device)
input_text = "Early to bed and early to rise makes a man wealthy, healthy and wise."

batch = pegasus_tokenizer([input_text], truncation=True, padding='longest', max_length=60, return_tensors="pt").to(torch_device)
translated = pegasus_model.generate(**batch, max_length=60, num_beams=BEAM_NUM, num_return_sequences=RETURN_SEQ_NUM, temperature=1.5)
tgt_text = pegasus_tokenizer.batch_decode(translated, skip_special_tokens=True)
for each_text in tgt_text:
print(each_text)
```

Original Text: 'Early to bed and early to rise makes a man healthy, wealthy and wise'
Augmented Text:
A man who is wealthy, healthy and wise can be found early to bed and early to rise.
A man who sleeps early and rises early is wealthy, healthy and wise.
A man is wealthy, healthy and wise if he sleeps early and rises early.
A man is wealthy, healthy and wise if he sleeps early.
A man is wealthy, healthy and wise when he is asleep and awake.
A man is wealthy, healthy and wise if he is up early.
A man is wealthy, healthy and wise if he is up early to bed and early to rise.
A man is wealthy, healthy and wise when he is awake early.
A man is wealthy, healthy and wise if he is awake early to bed and early to rise.
A man is wealthy, healthy and wise if he is awake early.

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 47-56
Published Online June 2023 in IJEAST (http://www.ijeast.com)

IV. NLP AUGMENTATION LIBRARY COMPARISON

Table 2. Library Comparison of NLP Augmentation

Library	Synonym Replacement	Word Insertion/Substitution/Deletion	Punctuation Insertion	Character Insertion	Augmentation using pre trained models
TextAttack	Yes	Yes	Yes	Yes	Yes
NLPAug	Yes	Yes	Yes	Yes	Yes
Googletrans	No	No	No	No	uses Google Translate Ajax API
TextAugment	Yes	Yes	Yes	Yes	Yes
AugLy	Yes	Yes	Yes	Yes	No
Parrot Paraphraser	No	No	No	No	Yes
Pegasus Paraphraser	No	No	No	No	Yes

V. CONCLUSION:

Data Augmentation in the field of NLP is widely used when there is scarcity of data. This survey study will help to know the basics of text data augmentation as well as the tools available for it and how to use it from its installation to implementation. In this study analysis table on all the tool is also present for anyone to decide which library to select as per their requirement. Implementation of the libraries feature with examples is done which will help for better understanding of features present. Hereafter, more analysis on it can be done to provide more deep knowledge about other tools available in market. Some standard datasets can also be taken and evaluate performance of each tool and analyze it's result.

VI. REFERENCES

[1]. Data Augmentation In NLP, https://towards data science.com/data-augmentation-in-nlp-2801a34dfc28

[2]. Text Data Augmentation in Natural Language Processing with Texattack, https://www.analyticsvidhya.com/blog/2022/02/text-data-augmentation-in-natural-language-processing-with-texattack/#:~:text=Data%20Augmentation%20(DA)%20Technique%20is,performance%20in%20the%20classification%20task.

[3]. 5 Data Augmentation Techniquesfor TextClassification, https://saurabhk30.medium.com/5-data-augmentation-techniques-for-text-classification-d14f6d8bd6aa

[4]. Data Augmentationfor NLP,https://blog.paperspace.com/data-augmentation-for-nlp/

[5]. Howtoperform Data Augmentationin NLP Projects,https://freecodecamp.org/news/how-to-perform-data-augmentation-in-nlp-projects/

[6]. Data Augmentation NLP, https://neptune.ai/blog/data-augmentation-nlp

[7]. Text Data Augmentation in natural language processing with Texattack, https://analyticsvidhya.com/blog/2022/02/text-data-augmentation-in-natural-language-processing-with-texattack/

[8]. nlpaug-A Pythonlibrarytoaugmentyourtextdata, https://www.analyticsvidhya.com/blog/2021/08/nlpaug-a-python-library-to-augment-your-text-data/

[9]. Text Data Augmentation,https://towardsdatascience.com/text-data-augmentation-f4143571ecd2

[10]. Howtouseauglyonimage, video, audioandtext,https://analyticsarora.com/how-to-use-augly-on-image-video-audio-and-text/

[11]. NLP Data Augmentation,https://pemagrg.medium.com/nlp-data-augmentation-a346479b295f

[12]. ParrotParaphraser, https://github.com/PrithivirajDamodaran/Parrot_Paraphraser

[13]. Pegasus Paraphrase, https://huggingface.co/tuner007/pegasus_paraphrase

[14]. Jingqing Zhang, Yao Zhao, Mohammad Saleh, Peter J. Liu: PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization, https://arxiv.org/abs/1912.08777

[15]. Steven Y. Feng, Varun Gangal, Jason Wei, SarathChandar, Soroush Vosoughi, Teruko Mitamura, Eduard Hovy: A Survey of Data Augmentation Approaches for NLP https://arxiv.org/pdf/2105.03075v5.pdf

[16]. Connor Shorten, Taghi M. Khoshgoftaar and BorkoFurht: Text Data Augmentation for Deep Learning https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00492-0