Question Generation from Fable Stories

PHASE 2

Team - DoNut Give Up

Gargi Shroff
Swarang Joshi
Ketaki Shetye

Project Timeline

PHASE 1

Presenting the general outline of the project and procedure to be followed.

PHASE 2

Data Preprocessing and Text Analysis

PHASE 3

T.B.D.

Data Pre-Processing

Steps involved in preprocessing are as follows -

1. Scraping the required text from the collected Dataset.
   Link of the resource for the Dataset - https://www.hindisahityadarpan.in/2016/06/panchatantra-complete-stories hindi.htm
2. Removing the Punctuation Marks from the Dataset.
3. POS Tagging of the obtained Dataset
4. Chunking

Removing Punctuation Marks

```
class MyClass:
    def __init__(self, text):
        self.text = text

    def clean_text(self):
        text = self.text
        text = re.sub(r'(\d+)',r'',text)
        text = text.replace(u'!','')
        text = text.replace(u',','')
        text = text.replace(u'"','')
        text = text.replace(u'(','')
        text = text.replace(u')','')
        text = text.replace(u'"','')
        text = text.replace(u':','')
        text = text.replace(u"'",'')
        text = text.replace(u"‘",'')
        text = text.replace(u"’",'')
        text = text.replace(u"''",'')
        text = text.replace(u".",'')
        text = text.replace(u"!",'')
        text = text.replace(u'“','')
        text = text.replace(u'”','')
        self.text = text

my_obj = MyClass(text)
my_obj.clean_text()
cleaned_text = my_obj.text
print(cleaned_text)
```

Punctuation marks are not always used consistently across different types of text data. For example, the use of commas, periods, and other punctuation marks can vary depending on the writer or the context. Removing punctuation marks helps to standardize the text data and makes it easier to compare and analyze different datasets.

Parts of Speech Tagging

```
[5]  nltk.download('indian')

     [nltk_data] Downloading package indian to /root/nltk_data...
     [nltk_data]   Unzipping corpora/indian.zip.
     True
```

```
[7]  from nltk.corpus import indian
     from nltk.tag import tnt
```

```
[24] def hindi_model():
         train_data = indian.tagged_sents('hindi.pos')
         tnt_pos_tagger = tnt.TnT()
         tnt_pos_tagger.train(train_data)
         return tnt_pos_tagger

     model = hindi_model()
     new_tagged = [(model.tag(nltk.word_tokenize(cleaned_text)))]
     print(new_tagged)
```

POS tagging is the process of assigning a part of speech tag to each word in the text data. This helps to identify the role of each word in a sentence, which is essential, because it helps to identify the grammatical structure of sentences and the roles played by different words within them.

Chunking

```
# Define grammar rules for chunking
grammar = r"""
    NP: {<NN.*|PRP.*|VM.*|NST.*|PR.*|QC.*|JJ.*>}              # Noun phrases
    PP: {<PSP.*|PRL.*><NP>}                                   # Prepositional phrases
    VP: {<VM.*><NP|PP>*}                                      # Verb phrases
    ADJP: {<JJ.*><RP.*>*}                                    # Adjective phrases
    QP: {<QC.*><RP.*>*}                                      # Quantity phrases
    RB: {<RB.*>}                                             # Adverbs
    PP: {<PSP.*|PRL.*><NP>}                                  # Postpositions followed by noun phrases
    NP: {<NP><PP>}                                           # Noun phrases followed by prepositional phrases

    FV: {<VM.*><VAUX.*>}                                      # Finite Verb chunk
    NFV: {<VM.*><PP|NP|ADJ|QF>*}                             # Non-finite Verb chunk
    IVC: {<VAUX.*><VM.*><PP|NP|ADJ|QF>*}                      # Infinitival Verb chunk
    VG: {<VM.*><VM.*><PP|NP|ADJ|QF>*}                         # Verb chunk (Gerund)
    ADJ: {<JJ.*><NP|PP>*}                                    # Adjectival chunk
    ADV: {<RB.*><NP|PP>*}                                    # Adverb chunk
    NEG: {<NEG.*><NP|PP|VP>}                                 # Chunk for negatives
    CONJ: {<CCP.*><NP|VP>}                                   # Conjuncts

    FRAG: {<.*>+}                                            # Chunk fragments
    MISC:
        {<SYM.*>}                                            # Symbols
        {<INTF.*>}                                           # Interjections
        {<RP.*>}                                             # Particles
"""

import nltk
from nltk.chunk import RegexpParser
from nltk.corpus import indian

chunk_parser = RegexpParser(grammar)

# Perform chunking on each sentence in the dataset
chunked_data = [chunk_parser.parse(sentence) for sentence in new_tagged]

# Print the chunked sentences
for chunked_sentence in chunked_data:
    print(chunked_sentence)
```

Chunking is the process of grouping together words in a sentence based on their syntactic structure. This helps to identify meaningful phrases or "chunks" of text, which can be used for identifying the key information, reducing ambiguity, and handling grammatical variations, chunking helps to improve the accuracy and quality of the generated text.

https://colab.research.google.com/drive/1GBxyUMcwzxwxDij6E_79BGjJipcek18u

THANK YOU FOR EXPLORING THE WORLD OF QUESTION GENERATION USING FABLES WITH US!