Getting Started with NLTK

NLTK (Natural Language Toolkit) is a powerful Python library for working with human language data. It provides easy-to-use interfaces to multiple text corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and more.

Task 1: Getting Ready

Firstly, ensure you have NLTK installed. If you haven't installed it yet, use pip (Python's package installer) in your command line:

```
pip install nltk
```

Browse the available packages using

```
nltk.download()
```

Download the book collections for the moment. Load them up using

```
from nltk.book import *
```

The corpus will be loaded as text1, text2, …. You can view these texts by giving text1 on the prompt to see their details. To search within a given text, use:

```
text1.concordance("monster")
```

The search above gives exact matching. To search for similar words, use:

```
text1.similar("monster")
```

Each time the queries are executed, it returns some additional text (left and right of the word) as context. We can search for words in terms of common context as:

```
text1.common_contexts(["monster", "person"])
```

To see the occurrence of a word in a text appears, how many times and where in a text, you can use dispersion plots (will not work in shell but in jupyter)

```
text1.dispersion_plot(["monster", "evil", "devil"])
```

To see all unique vocabulary in a given corpus, you can use:

```
set(text1)
```

Or to view the sorted list, just use:

```
sorted(set(text1))
```

Check the sizes of the vocabulary in the corpus using:

```
len(text3)
```

You can measure the lexical richness or lexical diversity of a corpus through as a ratio of unique words / total words, i.e. the higher, the better.

Using the above information, fill the following table:

Corpus	Text1	Text2	Text3	Text4	Text5	Text6	Text7	Text8	Text9
Corpus Name									

Corpus Length									
Unique Words									
Lexical Richness									

Task 2: Term Frequency

Similar to lexical richness, if you want to determine the count or frequency of a particular word appearing in a corpus, you can use:

```
text1.count('monster') / len(text1) * 100
```

Define a function **TF()** which can take the corpus and the token and return this value for you. In the case of log scales, you can use:

```
math.log(text1.count('monster')+1,10)
```

For this, use a function **LOGTF()** which can do the same.

For Inverse Document Frequency, you can use:

```
math.log(9 / text1.count('monster'), 10)
```

Prepare a function IDF() for the above.

Obtain the 10 most common words in a given corpus using the Frequency Distribution:

```
fdist1 = FreqDist(text1)

fdist1.most_common(3)
```

You would naturally want to use remove some stop words from the corpus (See Task 3). To view the most common words as a plot, use:

```
fdist1.plot(50)
```

Now, fill the table below:

	Text1		
Tokens	TF()	LOGTF()	IDF()
monster			
evil			
devil			
the			
Common word 1			
Common word 2			
Common word 3			

Task 3: Tokenization & POS

To try it out, try a simple tokenization task using the NLTK Punkt tokenizer.

```
nltk.download('punkt')

text = "NLTK is a powerful library for natural language processing."

words = nltk.word_tokenize(text)

sentences = nltk.sent_tokenize(text)

print(words)
print(sentences)
```

To to the text, assign parts of speech tags to the sentencs:

```
nltk.download('averaged_perceptron_tagger')

tags = nltk.pos_tag(words)

print(tags)
```

Remove some of the stop words like is, a, etc. in the sentence:

```
nltk.download('stopwords')

from nltk.corpus import stopwords

filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]

print(filtered_words)
```