```
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import string
import PyPDF2
# download stopwords and punkt
nltk.download('stopwords')
nltk.download('punkt')

# define stopwords and stemmer
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

# define function for text preprocessing
def preprocess_text(text):
    # convert to lowercase
    text = text.lower()
    # remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # tokenize text into words
    tokens = word_tokenize(text)
    # remove stopwords
    tokens = [word for word in tokens if not word in stop_words]
    # stem words
    tokens = [stemmer.stem(word) for word in tokens]
    # join tokens into a string
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text

# open the PDF file
pdf_file = open('TCD.pdf', 'rb')

# create a PDF reader object
pdf_reader = PyPDF2.PdfReader(pdf_file)

# extract text from the PDF file
text = ''
for page in range(len(pdf_reader.pages)):
    page_obj = pdf_reader.pages[page]
    text += page_obj.extract_text()

# preprocess the text
preprocessed_text = preprocess_text(text)

# close the PDF file
pdf_file.close()

# print the preprocessed text
print(preprocessed_text)

from sklearn.feature_extraction.text import CountVectorizer

# create a CountVectorizer object
vectorizer = CountVectorizer()

# fit the vectorizer to the preprocessed text data
term_document_matrix = vectorizer.fit_transform([preprocessed_text])
```

```
# get the feature names (terms)
feature_names = vectorizer.get_feature_names()

# print the term-document matrix and the feature names
print(term_document_matrix.toarray())
print(feature_names)

from sklearn.decomposition import TruncatedSVD

# create a TruncatedSVD object
svd = TruncatedSVD(n_components=2)

# apply SVD to the term-document matrix
svd.fit(term_document_matrix)

# get the transformed matrix (document embeddings)
document_embeddings = svd.transform(term_document_matrix)

# print the transformed matrix (document embeddings)
print(document_embeddings)

# get the left singular vectors
left_singular_vectors = svd.components_

# print the shape of the left singular vectors
print(left_singular_vectors.shape)

# loop through each topic and print the top 10 important terms
num_top_terms = 10
for i, topic in enumerate(left_singular_vectors):
    print("Topic {}: ".format(i+1))
    top_terms_idx = topic.argsort()[-num_top_terms:][::-1]
    top_terms = [feature_names[idx] for idx in top_terms_idx]
    print(", ".join(top_terms))
```