```
import numpy as np
import re
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize

# --------------------------------------------------------------------------------------------
---------------------------------------------------------

def get_files(n):
    file_list = []
    text = ""
    for i in range(1, n):
        file_string = "./Corpuses/corpus" + str(i) + ".txt"
        with open(file_string, "r", encoding='utf-8') as file_handle:
            text = file_handle.read()
            file_list.append(text)

    return file_list

# --------------------------------------------------------------------------------------------
---------------------------------------------------------

def clean_text(file_list):

    list_of_files = []
    stop_words = stopwords.words('english')
    vocab = []
    j=1
    for i in file_list:
        text_file = i.lower()
        tokens = word_tokenize(text_file)

        tokens_without_stopwords = [token for token in tokens if token not in stop_words and
token.isalpha()]
        # print(tokens_without_stopwords)
        text_without_stopwords = " ".join(tokens_without_stopwords)
        if j==1:
            vocab = tokens_without_stopwords
        else:
            vocab.append(tokens_without_stopwords)

        list_of_files.append(text_without_stopwords)
    vocab = set(vocab)
    return list_of_files, vocab

# --------------------------------------------------------------------------------------------
---------------------------------------------------------

def calculate_tf_idf(n):
    all_files = get_files(n)
    filtered_files, vocab = clean_text(all_files)

    tfidf = TfidfVectorizer()
    result = tfidf.fit_transform(filtered_files)
    avg_of_tfidf = np.mean(result, axis=0).tolist()[0]
    feature_names = tfidf.get_feature_names_out()
    word_tfidf_scores = list(zip(feature_names, avg_of_tfidf))
    sorted_word_tfidf_scores = sorted(word_tfidf_scores, key=lambda x: x[1], reverse=True)

    top_list = sorted_word_tfidf_scores[:40]
    top_dict = dict(top_list)

    return list(top_dict.keys())[:50], vocab

# --------------------------------------------------------------------------------------------
```

```
--------------------------------------------------------

def tfidf():

    top_words = calculate_tf_idf(21)
    return top_words

# ------------------------------------------------------------------------------------------
--------------------------------------------------------

def listoffiles():
    all_files = get_files(21)
    filtered_files, vocab = clean_text(all_files)
    return filtered_files, vocab

# ------------------------------------------------------------------------------------------
--------------------------------------------------------

def docsentokens():
    doc_dict = {}
    i = 1
    docs_list = get_files(21)
    for doc in docs_list:
        sent_no_text = "doc" + str(i)
        sentences_list = sent_tokenize(doc)
        doc_dict[sent_no_text] = sentences_list
        i+=1

    return doc_dict

# ------------------------------------------------------------------------------------------
--------------------------------------------------------

def filmography():

    with open("./Corpuses/corpus0.txt", "r") as file:
        text = file.read()
    text = re.sub(r'[â€–“”]', '-', text)
    text = re.sub('---', '-', text)
    text = re.sub('--', '', text)

    movie_section, tv_show_section = text.split("===")

    movies = list(movie_section.split('\n'))

    tvshows = list(tv_show_section.split('\n'))

    j = 0
    movies_dict = {}
    flag = 1

    for i in range(len(movies)):
        # print("i:", i)
        if movies[i].startswith('1') or movies[i].startswith('2') or movies[i] == 'TBA':
            year = movies[i]
            # print("\tYear:", year)
            movies_dict[year] = []
            # print("Year:", year)
            j = i + 1
            if (j > len(movies)):
                break
            # print('\tj value before while:', j, ', movies[j]:', movies[j])
            flag = 1
            while (flag == 1):
                if (j >= (len(movies))):
                    # print("\t\tj reached end of list, breaking loop")
                    break
                if movies[j].startswith('1') or movies[j].startswith('2') or
(movies[j].startswith('TBA')):
                    flag = 0
                    # print("j reached next year value, flag is 0, exiting while loop")
                    break

                if ((j - i) == 1):
                    movies_dict[year].append(movies[j])
                    # print("\t\tj-i=1:", j-i, ', appending value to dict:', movies[j])

                elif  (((j - i) % 2 )== 0):
                    movies_dict[year].append(movies[j])
                    # print("\t\tj-i%2=0:", j-i, ', appending value to dict:', movies[j])

                elif ((j - i) > 8):
                    flag = 0
                    # print("\t\tj-i>8:", j-i, '\n\t\tbreaking while loop')

                j += 1

            i = j - 1

    # print('\n\nMovies:')
    # for i in movies_dict.keys():
    #     print(i, ":", movies_dict[i])

    with open('./Corpuses/movies.pickle', 'wb') as handle:
        pickle.dump(movies_dict, handle, protocol=pickle.HIGHEST_PROTOCOL)

    j = 0
    tvshows_dict = {}
    flag = 1

    for i in range(len(tvshows)):
        # print("i:", i)
        if tvshows[i].startswith('1') or tvshows[i].startswith('2') or tvshows[i] == 'TBA':
            year = tvshows[i]
            # print("\tYear:", year)
            tvshows_dict[year] = []
            # print("Year:", year)
            j = i + 1
            if (j > len(tvshows)):
                break
            # print('\tj value before while:', j, ', tvshows[j]:', tvshows[j])
            flag = 1
            while (flag == 1):
                if (j >= (len(tvshows))):
                    # print("\t\tj reached end of list, breaking loop")
                    break

                if tvshows[j].startswith('1') or tvshows[j].startswith('2') or
(tvshows[j].startswith('TBA')):
                    flag = 0
                    # print("j reached next year value, flag is 0, exiting while loop")
                    break

                if ((j - i) == 1):
                    tvshows_dict[year].append(tvshows[j])
                    # print("\t\tj-i=1:", j-i, ', appending value to dict:', tvshows[j])

                elif  (((j - i) % 2 )== 0):
                    tvshows_dict[year].append(tvshows[j])
                    # print("\t\tj-i%2=0:", j-i, ', appending value to dict:', tvshows[j])

                elif ((j - i) > 8):
                    flag = 0
                    # print("\t\tj-i>8:", j-i, '\n\t\tbreaking while loop')

                j += 1

            i = j - 1

    # print("\n\nTV Shows:")
    # for i in tvshows_dict.keys():
    #     print(i, ":", tvshows_dict[i])

    with open('./Corpuses/tvshows.pickle', 'wb') as handle:
        pickle.dump(tvshows_dict, handle, protocol=pickle.HIGHEST_PROTOCOL)

    return movies_dict, tvshows_dict

# ---------------------------------------------------------------------------------------------
--------------------------------------------------------

# print("\n\nTF-IDF list of words: \n", tfidf())

print(docsentokens())
```