Prototype 2

1. Preliminary Data Preperation

1.1 Data Import and Cleaning

The data cleaning and data reshape portions will not vary much from the Feature Prototype (FP), since the entire dataset is similarly formatted to the test dataset used in the FP.

One minor change is that we have chosen to remove the Computer Science Wikipedia Articles (CSAI) category from the HC3 dataset. This is as this category contains many wikipedia formatting tags that are difficult to remove, as they have no standardised format between tags. The CSAI category makes up an very small portion (~3%) of the entire dataset, and we believe that its removal is very unlikely to skew the dataset in a meaningful way, and it is safe to remove.

First, install the neccesary libraries and import modules

In [ ]:
```
# !pip install pandas
# !pip install numpy
# !pip install textblob
# !pip install nltk
```

In [ ]:
```
import pandas as pd
import json
```

Read the input jsonl file to an intermediate dataframe.

This time, we will read the entire HC3 dataset

In [ ]:
```
# convert jsonl to dataframe
with open('all.jsonl', encoding='utf8') as f:
    # read lines from jsonl into an intermediate df
    lines = f.read().splitlines()
    df_inter = pd.DataFrame(lines)
    df_inter.columns = ['json_element']
```

In [ ]:
```
df_inter.sample(3)
```

Out[ ]:

	json_element
12552	{"question":"What is that noise your computer ...
5413	{"question":"How are Nuclear Missiles Safely D...
20898	{"question":"Finding a good small business CPA...

Next, split the dataframe into its respective columns

In [ ]:
```
#normalise the intermediate dataframe to seperate the data into columns
df = pd.json_normalize(df_inter['json_element'].apply(json.loads))
```

In [ ]:
```
df.sample(3)
```

Out[ ]:

	question	human_answers	chatgpt_answers	index	source
6296	Do dogs understand concepts like fish need wat...	[That dog is not trying to save the fish . He ...	[Dogs are intelligent animals, but it is unlik...	NaN	reddit_eli5
22213	Is it advisable to go for an auto loan if I ca...	[What percentage of your savings is the full c...	[It is generally advisable to pay for a car in...	NaN	finance
19976	Can everyday people profit from unexpected wor...	[In the UK there are spread betting firms (ess...	[It is possible for everyday people to profit ...	NaN	finance

Due to formatting issues (this category has a large amount of wikipedia formatting tags that vary greatly in format), we will also drop any rows that belong to the Computer Science Wikipedia Articles category (842 rows).

In [ ]:
```
prev_len = len(df) #find the current length of the dataframe before removal
prev_len
```

Out[ ]: 24322

In [ ]:
```
df = df[df.source != 'wiki_csai'] #remove the wiki_csai rows
df.reset_index(inplace=True, drop=True) #also reset the indexes of the dataframe
```

In [ ]:
```
cur_len = len(df)
prev_len - cur_len #check to make sure the correct number of rows was removed
```

Out[ ]: 842

In [ ]:
```
df.sample(3)
```

Out[ ]:

	question	human_answers	chatgpt_answers	index	source
4981	How do Mountains affect the weather ? I live a...	[Air that used to be at sea level changes its ...	[Mountains can have a big impact on the weathe...	NaN	reddit_eli5
410	why is n't happy hour one single hour ? Why is...	[Because it is hard to drink enough in one hou...	[Happy hour is called happy hour because it is...	NaN	reddit_eli5
13963	Why are the police : military of totalitarian ...	[There are two psychological terms , " ingroup...	[It's important to note that not all police of...	NaN	reddit_eli5

Now that we have removed the wiki_csai category, we can also drop the 'index' and 'source' columns, as they will not be used.

In [ ]:
```
df.drop(['index', 'source'], axis=1, inplace=True) # drop index and source columns
```

Next, we can inspect the data for missing values.

In [ ]:
```
# find any missing/NA values
df.isna().sum()
```

Out[ ]:
```
question           0
human_answers      0
chatgpt_answers    0
dtype: int64
```

No missing values were found.

Since each answer is a list, a list containing just empty strings, or an empty list will not be considered by panda's isna() function as a NA value. Thus, we will also inspect the dataframe for empty strings.

Some empty strings were found. Since these empty strings make up an insignificant portion of the dataset (~2%), they can be removed without significantly changing the makeup of the dataset.

In [ ]:
```
#check for empty strings
drop_cntr = 0
for i in range(len(df)):
    human_ans = df.at[i, 'human_answers']
    gpt_ans = df.at[i, 'chatgpt_answers']
    # if the answers are an empty string, or an empty list,
    if '' in human_ans or '' in gpt_ans or len(human_ans) == 0 or len(gpt_ans) == 0
        #drop the row
        df.drop(i, axis=0, inplace=True)
        drop_cntr += 1
#print drop counter
print(f'dropped {drop_cntr} rows')
```

```
dropped 470 rows
```

Next, reset the indexes of the dataset without the removed rows (so that it can be iterated over later)

In [ ]:
```
df.reset_index(inplace=True)
df.drop(df.columns[0], axis=1, inplace=True)
```

We will describe the dataset to inspect the values contained within

In [ ]:
```
df.describe(include='all')
```

Out[ ]:

	question	human_answers	chatgpt_answers
count	23010	23010	23010
unique	22482	21508	22447
top	Why we learned to cover our private parts with...	[The attack on Pearl Harbor (called Hawaii Ope...	[!Only one message at a time. Please allow any...
freq	2	3	14

We have also found some duplicate values, including duplicate questions, human answers, and chatgpt answers (which seems to be an error message)

These rows will be removed once the dataframe has been processed further (as it will be easier to remove them after the dataframe is converted to strings)

Next, we will inspect the dataframe's data types

In [ ]:
```
# check datatypes
df.dtypes
```

Out[ ]:
```
question           object
human_answers      object
chatgpt_answers    object
dtype: object
```

The dataframe will be converted to the string datatype.

In [ ]:
```
#convert to string
df = df.astype(str)
```

Next, we will need to strip the trailing/leading square brackets from the answers, and remove any newline characters (\n). We will also remove some Reddit text formatting tags and syntaxes.

In [ ]:
```
for i in range(len(df)):
    human_ans = df.at[i, 'human_answers']
    # strip leading/trailing square brackets and quotation marks
    human_ans = human_ans.lstrip('[\'\"')
```

```
    human_ans = human_ans.rstrip(']\'\"')
    human_ans = human_ans.replace('\\n', "") # remove all newline chars
    human_ans = human_ans.replace('\\r', "") # remove all carraige return chars
    human_ans = human_ans.replace('*', "") # remove reddit formatting *
    human_ans = human_ans.replace('[', "") # remove reddit formatting square bracke
    human_ans = human_ans.replace(']', "")
    df.at[i, 'human_answers'] = human_ans

    gpt_ans = df.at[i, 'chatgpt_answers']
    # strip leading/trailing square brackets and quotation marks
    gpt_ans = gpt_ans.lstrip('[\'\"')
    gpt_ans = gpt_ans.rstrip(']\'\"')
    gpt_ans = gpt_ans.replace('\\n', "") # remove all newline chars
    gpt_ans = gpt_ans.replace('\\r', "") # remove all carraige return chars
    gpt_ans = gpt_ans.replace('*', "") # remove reddit formatting *
    gpt_ans = gpt_ans.replace('[', "") # remove reddit formatting square brackets
    gpt_ans = gpt_ans.replace(']', "")
    df.at[i, 'chatgpt_answers'] = gpt_ans
```

In [ ]:
```
df.describe(include='all')
```

Out[ ]:

	question	human_answers	chatgpt_answers
count	23010	23010	23010
unique	22482	21508	22445
top	Why we learned to cover our private parts with...	The attack on Pearl Harbor (called Hawaii Oper...	!Only one message at a time. Please allow any ...
freq	2	3	17

Now that the dataframe has been converted to strings, we can easily remove the rows containing the chatgpt error messages

In [ ]:
```
df = df[df['chatgpt_answers'].str.contains('There was an error generating a respons
```

We will also remove the duplicate questions and answers.

In [ ]:
```
df.drop_duplicates(subset = ['human_answers'], inplace=True)
df.drop_duplicates(subset = ['chatgpt_answers'], inplace=True)
df.drop_duplicates(subset = ['question'], inplace=True)
```

In [ ]:
```
df.describe(include ='all')
```

Out[ ]:

	question	human_answers	chatgpt_answers
count	21398	21398	21398
unique	21398	21398	21398
top	Why is every book I hear about a " NY Times # ...	Basically there are many categories of " Best ...	There are many different best seller lists tha...
freq	1	1	1

Additionally, the HC3 dataset covers both Chinese and English answers. Though we only imported the English portion of the dataset, it would be advisable to ensure all answers are actually in English.

This check will be carried out using the langdetect python package, which is a Python port of Google's language detection library.

First, install and import langdetect

In [ ]:
```
#!pip install langdetect
```

In [ ]:
```
from langdetect import detect
```

Next, define a function to detect if the text is english.

In [ ]:
```
def detect_english(answer):
    # detects if the given answer is in English
    if (detect(answer) == 'en') == False:
        raise Exception() #detects if the
```

We will also reset the indexes of the dataframe so we can iterate through it

In [ ]:
```
df.reset_index(inplace=True)
```

Call the detect_english method on both the human and chatgpt answers to remove any non-English answers. Once again, this makes up an extremely small portion of the entire dataset

In [ ]:
```
drop_cntr = 0 # counter for number of rows dropped
for i in range(0, len(df)):
    # extract human and gpt answers
    human_ans = df.at[i, 'human_answers']
    gpt_ans = df.at[i, 'chatgpt_answers']

    try:
        # try to call detect_english on both answers
        detect_english(human_ans)
        detect_english(gpt_ans)
    except:
        # print error message if either answer raises an exception
        print(f'non-english answer at f{i}')
```

```
        # drop the row
        df.drop(i, axis=0, inplace=True)
        drop_cntr += 1

# print total rows dropped
print(f'dropped {drop_cntr} rows')
```

```
non-english answer at f15299
non-english answer at f15520
non-english answer at f15878
non-english answer at f15908
non-english answer at f17052
non-english answer at f20613
non-english answer at f20659
non-english answer at f20681
dropped 8 rows
```

This results in a cleaned dataset that is about 88% the size of the original dataset. In our opinion, the majority of the original dataset is preserved, and the cleaning is unlikely to significantly affect our results.

1.2 Data Reshape

We will also need to reshape the dataframe to seperate human and chatgpt answers into distinct rows.

First, we will seperate the dataframe into 2 seperate dataframes, each containing only 1 type of answer.

We will also add an additional 'is_human' column to each dataframe, indicating if the answer was generated by a human

In [ ]:
```
# retrieve human answers
df_human = df[['question', 'human_answers']].copy() #copy the question and human an
df_human['is_human'] = 1 #set a column indicating if the answer is human
df_human.rename(columns = {'human_answers':'answer'}, inplace = True) # rename colu
df_human.sample(2)
```

Out[ ]:

	question	answer	is_human
7619	Why do people close their eyes when they hear ...	Loud noises usually mean explosions or alike ,...	1
2941	What is DNS and DHCP . I have no idea . Please...	DNS is like a phone book . You either know...	1

In [ ]:
```
# do the same for chatgpt answers
df_gpt = df[['question', 'chatgpt_answers']].copy()
df_gpt['is_human'] = 0
df_gpt.rename(columns = {'chatgpt_answers':'answer'}, inplace = True)
df_gpt.sample(2)
```

Out[ ]:

	question	answer	is_human
5069	Why ca n't you use laptop parts for a PC ? I k...	\While it is possible to use some laptop parts...	0
10736	What is this new " Common Core " math ? I feel...	Common Core" is a set of standards for math an...	0

Next, both dataframes will be concatenated to join them together into a cleaned dataframe

In [ ]:
```
# concat both dataframes
df_cleaned = pd.concat([df_human, df_gpt])
df_cleaned.sample(3)
```

Out[ ]:

	question	answer	is_human
3146	What can high - profile law firms do to win ca...	High-profile law firms often have more resourc...	0
6474	Do Cubans get to pick their job ? Communism me...	In a communist system, the government typicall...	0
9653	Evolution Please explain to me Evolution , the...	Basically it is very Very simple . So simple t...	1

We will also use the panda's value_counts function to ensure that there is the expected number of each type of answer

In [ ]:
```
# sum is_human to ensure there is a 50:50 split of answers
df_cleaned['is_human'].value_counts()
```

Out[ ]:
```
is_human
1    21390
0    21390
Name: count, dtype: int64
```

Lastly, we will shuffle the order of the answers

In [ ]:
```
# shuffle human and chatgpt generated answers
df_cleaned = df_cleaned.sample(frac = 1).reset_index(drop=True)
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	question	answer	is_human
32147	what happened on the moon during the period of...	During this event a very large number of the i...	1
17350	The causes of WWI . Other than MANIA , I 'm so...	Franco - Russian alliance : France angry over ...	1
31871	why does ice freeze hard but ice cream freeze ...	Ice cream is made with cream, which is a liqui...	0

1.3 Lemmatisation

We will also create both a lemmatised, and a filtered (both lemmatised, and stopword-removed) version of each answer for later usage in some methods, such as sentiment analysis.

Compared to stemming, lemmatisation preserves the context and meaning of a word whe converting it to its base form. Preserving the context and meaning of each word will allow for a more accurate sentiment analysis

First, download NLTK packages, and import the required modules.

In [ ]:
```
# Also download NLTK packages
nltk.download('stopwords')
nltk.download('punkt')

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
```

Next, initialise the lemmatiser, and the corpus of stopwords. The stopword corpus used will be NLTK's general English stopword corpus.

In [ ]:
```
# init lemmatiser and stopwords
lemmatizer = WordNetLemmatizer()
stopwords = stopwords.words('english')

# also create empty columns for filtered/lemmatised answers
df_cleaned['filtered_answer'] = ""
df_cleaned['lemmatised_answer'] = ""
```

Next, iterate through each answer, processing each answer by either:

1. just lemmatising the answer, or
2. both removing stopwords and lemmatising the answer.

In [ ]:
```
for i in range(len(df_cleaned)):

    answer = df_cleaned.at[i, 'answer'] #retrieve answer text from df
```

```
    tokens = word_tokenize(answer) # tokenise the answer

    # lemmatise answer
    lemmatised_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    # reconstruct the answer as a string, and store lemmatised answer in column
    df_cleaned.at[i, 'lemmatised_answer'] = ' '.join(lemmatised_tokens)

    # remove stopwords
    tokens_no_stopwords = [token for token in tokens if token not in stopwords]
    # lemmatise stopword-removed answer
    lemmatised_tokens_no_stopwords = [lemmatizer.lemmatize(token) for token in toke

    # reconstruct the answer as a string, and store filtered answer in column
    df_cleaned.at[i, 'filtered_answer'] = ' '.join(lemmatised_tokens_no_stopwords)
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	question	answer	is_human	filtered_answer	lemmatised_answer
21766	What does it mean for issuing corporations to ...	When markets are "frothy," it means that inves...	0	When market `` frothy , '' mean investor willi...	When market are `` frothy , '' it mean that in...
24652	How do Key Generators work and why are n't gam...	This is actually two separate things , key gen...	1	This actually two separate thing , key gen n\'...	This is actually two separate thing , key gen ...
1890	Who makes malware , viruses , and trojans ? Wh...	Pretty much anyone with a knowledge of compute...	1	Pretty much anyone knowledge computer programm...	Pretty much anyone with a knowledge of compute...

2. Feature Extraction

2.1 Basic Textual Features

We will extract a similar set of features to the ones used by Fröhling and Zubiaga [2021] - word count, sentence count, character count, average word length, and average sentence length.

In [ ]:
```
# import sentence and word tokenisers
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
```

In [ ]:
```
#initialise columns for the extracted features
df_cleaned['word_count'] = 0
df_cleaned['sentence_count'] = 0
df_cleaned['char_count'] = 0
```

```
df_cleaned['avg_sentence_len'] = 0
df_cleaned['avg_word_len'] = 0
```

Next, iterate through each answer, extracting the selected features.

In [ ]:
```
for i in range(len(df_cleaned)):
    answer = df_cleaned.at[i, 'answer'] # extract answer text from df

    ans_sent_token = sent_tokenize(answer)# tokenise answer into sentences
    sentence_count = len(ans_sent_token)# calculate sentence count
    df_cleaned.at[i, 'sentence_count'] = sentence_count

    ans_word_token = word_tokenize(answer) # tokenise answer into words
    word_count = len(ans_word_token)  # calculate word count
    df_cleaned.at[i, 'word_count'] = word_count

    char_count = len(answer) # calculate character count
    df_cleaned.at[i, 'char_count'] =  char_count

    df_cleaned.at[i, 'avg_sentence_len'] = word_count/sentence_count # calculate av
    df_cleaned.at[i, 'avg_word_len'] =  char_count/word_count   # calculate average
```

In [ ]:
```
df_cleaned.sample(2)
```

Out[ ]:

	question	answer	is_human	filtered_answer	lemmatised_answer	word_count
8632	Why is the phrase " Fuck you " so gravely offe...	The phrase "fuck you" is considered a very rud...	0	The phrase `` fuck '' considered rude offensiv...	The phrase `` fuck you '' is considered a very...	154
31332	Do I owe taxes if my deductions are higher tha...	In your case, I believe the answer is that you...	1	In case , I believe answer don\'t owe tax , de...	In your case , I believe the answer is that yo...	82

2.2 Sentiment Analysis

4 metrics will be assessed – negativity, neutrality, subjectivity, and overall polarity.

To calculate overall polarity, we will use 2 analysers - NLTK's VADER and TextBlob. This will allow us to obtain a secondary opinion, and provide a more accurate compound score.

The rest of the metrics will be extracted using only 1 analyser, as each analyser supports a different set of metrics. Negativity and neutrality will be extracted using VADER, and subjectivity will be extracted by Textblob.

First, import the required modules and download the required VADER NLTK package.

In [ ]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
```

In [ ]:
```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[ ]: True

We will also initialise empty columns for the extracted features.

In [ ]:
```
# initialise columns for the extracted features
df_cleaned['polarity'] = 0
df_cleaned['negativity'] = 0
df_cleaned['neutrality'] = 0
df_cleaned['subjectivity'] = 0
```

Next, initialise the NLTK VADER sentiment analyser (TextBlob does not need to be initialise using a constructor)

In [ ]:
```
analyser = SentimentIntensityAnalyzer()
```

Next, iterate through each filtered answer (lemmatised + stopwords removed), extracting the negativity (NLTK only), neutrality (NLTK only), polarity (both), and subjectivity (TextBlob only) of each answer.

In [ ]:
```
for i in range(len(df_cleaned)):
    answer = df_cleaned.at[i, 'filtered_answer']
    #analyse filetered answers
    nltk_scores = analyser.polarity_scores(answer)
    textblob_scores = TextBlob(answer).sentiment

    # extract neg and neu using nltk
    df_cleaned.at[i, 'negativity'] = nltk_scores['neg']
    df_cleaned.at[i, 'neutrality'] = nltk_scores['neu']

    # calculate average polarity by using results from both analysers
    df_cleaned.at[i, 'polarity'] = (nltk_scores['compound'] + textblob_scores.polar

    # extract subjectivity using textblob
    df_cleaned.at[i, 'subjectivity'] = textblob_scores.subjectivity
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	question	answer	is_human	filtered_answer	lemmatised_answer	word_count
37525	Whats the difference between a qualified and a...	Yes, as long as you write a call against your ...	1	Yes , long write call stock strike price great...	Yes , a long a you write a call against your s...	387
3863	Why does my body produce so much mucus when I ...	Your body produces mucus from mucous membranes...	1	Your body produce mucus mucous membrane presen...	Your body produce mucus from mucous membrane w...	210
8800	Why do hospitals have religious sounding names...	Hospitals often have religious sounding names ...	0	Hospitals often religious sounding name founde...	Hospitals often have religious sounding name b...	188

2.3 Syntactic and Lexical Density

2.3.1 Lexical Density

First, we will extract the lexical density of each answer. This will involve counting the number of lexical terms in each answer (nouns/proper nouns, adjectives, verbs, prepositions, and adverbs), and dividing them by the total number of words in the answer.

First, install and import spaCy, and download the pre-trained en_core_web_sm model, which is optimised for CPU processing.

In [ ]:
```
#!pip install spacy
#!python -m spacy download en_core_web_sm
```

In [ ]:
```
import spacy
```

Also, create an empty column for the lexical density feature

In [ ]:
```
df_cleaned['lexical_density'] = 0 # create an empty column for lexical density
```

Load the spacy en_core_web_sm model, and a list of POS tags that denote lexical words

In [ ]:
```
nlp = spacy.load("en_core_web_sm") # load spacy model
tags = ['PROPN', 'ADJ', 'NOUN', 'VERB', 'ADP', 'ADV'] # load list of desired lexica
```

Finally, iterate through every row and count the number of lexical words. Then, divide the number of lexical words by the total word count to calculate the lexical density of each

answer.

In [ ]:
```
for i in range(len(df_cleaned)):

    answer = df_cleaned.at[i, 'answer'] #extract the answer text
    word_count = df_cleaned.at[i, 'word_count'] #extract word count

    lexical_word_cntr  = 0 #initialise counter for lexical words
    ans_pos = nlp(answer) # analyse answer text using model

    # for each word in the analysed answer
    for token in ans_pos:
        # check if it is a lexical word (POS tag is in the list of desired tags)
        if token.pos_ in tags:
            lexical_word_cntr += 1

    df_cleaned.at[i, 'lexical_density'] = lexical_word_cntr/word_count # calculate
```

2.3.2 Named Entity Count

Next, we will extract the named entity features. The count of named entities will be extracted using spaCy.

In [ ]:
```
df_cleaned['NE_count'] = 0
```

In [ ]:
```
for i in range(len(df_cleaned)):
    answer = df_cleaned.at[i, 'answer'] #extract the answer text
    ans_NE = nlp(answer) # analyse answer text using the spaCy model
    df_cleaned['NE_count'] = len(ans_NE.ents) #retrieved the NE count for each answ
```

In [ ]:
```
df_cleaned.sample(1)
```

Out[ ]:

	question	answer	is_human	filtered_answer	lemmatised_answer	word_count
12654	What causes light headedness, nausea and fever...	There could be a number of potential causes fo...	0	There could number potential cause lightheaded...	There could be a number of potential cause for...	163

Since this feature takes a significant time to extract, we will save the current data to a csv file.

This avoids the need to re-run the entire code again when refreshing the jupyter kernel/restarting the notebook.

In [ ]:
```
df_cleaned.to_csv('intermediate_data_1.csv', mode='w')
```

2.3.3 Coreference Cluster Features

Next, we will also extract the Coreference Cluster features - average coreference cluster length, and number of coreference clusters.

This will be extracted using a pre-trained en_coreference_web_trf model from SpaCy's experimental package.

First, we will need to install spacy-experimental, and download the pre-trained coreference cluster model.

We will also install the tqdm package to print out a progress bar, so we can visualise the progress of the coreference cluster feature extraction (as coreference cluster extraction takes roughly 11-12 hours for the entire dataset)

First, install the required packages and import modules

In [ ]:
```
#!pip install spacy-experimental
#!pip install https://github.com/explosion/spacy-experimental/releases/download/v0.
#!pip install tqdm
```

In [ ]:
```
import spacy
import spacy_transformers
from tqdm import tqdm
```

Load the pretrained model, and read in the data from the intermediate csv file we saved earlier

In [ ]:
```
coref = spacy.load("en_coreference_web_trf")
```

```
c:\Users\Admin\OneDrive - Microsoft 365\Desktop\Prototype 2\.venv\Lib\site-packages
\spacy\util.py:910: UserWarning: [W095] Model 'en_coreference_web_trf' (3.4.0a2) was
trained with spaCy v3.3.0 and may not be 100% compatible with the current version
(3.6.1). If you see errors or degraded performance, download a newer compatible mode
l or retrain your custom model with the current spaCy version. For more details and
available updates, run: python -m spacy validate
  warnings.warn(warn_msg)
```

In [ ]:
```
df_cleaned = pd.read_csv('intermediate_data_1.csv')
```

Also, create empty columns for the coreference cluster features

In [ ]:
```
 # create empty columns for the coreference features
df_cleaned['coref_count'] = 0
df_cleaned['avg_coref_len'] = 0
```

Next, iterate over all rows in the dataframe, and extract the coreference spans (clusters).

This will allow us to count the number of coreference clusters, and calculate the average length of each cluster,

In [ ]:
```
for i in tqdm(range(0, len(df_cleaned))):
        answer = df_cleaned.at[i, 'answer'] #retrieve answer from df
        doc = coref(answer) # analyse answer using pretrained coref model

        coref_count = len(doc.spans) #get number of coref spans (clusters)

        # skip answer if no coref spans detected, since no clusters results in a 0
        # avoids div by 0 error
        if coref_count == 0:
            continue

        # initialise counter for total length of all coref spans
        total_span_len = 0
        # iterate over all spans, and add their length to the counter
        for span in doc.spans:
            total_span_len += len(doc.spans[span])

        avg_span_len = total_span_len/coref_count #calcuate average span length

        # set values in df
        df_cleaned.at[i, 'avg_coref_len'] = avg_span_len
        df_cleaned.at[i, 'coref_count'] = coref_count
```

```
  3%|▌          | 1074/42780 [18:40<9:24:31,  1.23it/s] Token indices sequence lengt
h is longer than the specified maximum sequence length for this model (527 > 512). R
unning this sequence through the model will result in indexing errors
100%|██████████| 42780/42780 [12:31:01<00:00,  1.05s/it]
```

Once again, we will save this to an intermediate csv file to avoid having to rerun this extraction again.

In [ ]:
```
df_cleaned.to_csv('intermediate_data_2.csv', mode='w')
```

2.4 Frequency

We will also be extracting 3 frequency features:

1. Zipfian distribution difference
2. Unique word count
3. Stop word count

2.4.1 Zipfian Difference

We will extract the difference between the text's Zipfian distribution line, and the Zipf's law line. (which we will refer to as Zipfian Difference)

First, install matplotlib, which will be used to assist in visualising this method.

Also, import the neccessary modules

In [ ]:
```
# !pip install matplotlib
```

In [ ]:
```
import string
import collections
import pandas as pd
import nltk
from nltk import word_tokenize
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm
```

We will also read in the csv file saved in the last step

In [ ]:
```
df_cleaned = pd.read_csv('intermediate_data_2.csv')
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	Unnamed: 0.1	Unnamed: 0	question	answer	is_human	filtered_answer	lemmat
22282	22282	22282	What causes metallic taste in mouth, shortness...	There are several possible causes for metallic...	0	There several possible cause metallic taste mo...	The poss
18460	18460	18460	How do physicists entangle particles in quantu...	Quantum entanglement is a phenomenon that occu...	0	Quantum entanglement phenomenon occurs two par...	entai phen
40163	40163	40163	How often do typical investors really lose money?	It is difficult to determine the frequency wit...	0	It difficult determine frequency typical inves...	It c fr

3 rows × 23 columns

In order to better illustrate this method, we will show the entire process on a random sample answer, before applying it to the entire text.

To start, we will extract the lemmatised answer (inclusive of stopwords) from the dataframe, and strip its punctuation. The lemmatised answer is used to reduce each word to its base form for counting (e.g., jump, jumped, jumping should all be reduced to 'jump' for counting)

In [ ]:
```
answer = df_cleaned.at[np.random.randint(0, high = len(df_cleaned)), 'lemmatised_an
answer = answer.translate(str.maketrans('', '', string.punctuation)) # strip punctu
```

Next, we will tokenise the answer, and convert it to a frequency distribution

In [ ]:
```
tokens = word_tokenize(answer) #tokenise answer
freq_dist = nltk.FreqDist(tokens) # create freqdist
freq_dist
```

Out[ ]:
```
FreqDist({'a': 8, 'to': 5, 'of': 5, 'the': 5, 'can': 4, 'economic': 4, 'recessio
n': 4, 'change': 4, 'in': 4, 'and': 4, ...})
```

Next, we will convert the frequency distribution to a ordered dictionary, sorted in descending order of word frequency.

This is so we can preseve the order of the dictionary, which will be used to plotting later on.

In [ ]:
```
freq_dict = dict([(m, n) for m, n in freq_dist.items()]) # create dictionary from f
freq_dict = collections.OrderedDict(dict(sorted(freq_dict.items(), key=lambda item:
print(freq_dict)
```

```
OrderedDict([('a', 8), ('to', 5), ('of', 5), ('the', 5), ('can', 4), ('economic',
4), ('recession', 4), ('change', 4), ('in', 4), ('and', 4), ('I', 3), ('is', 3), ('b
usiness', 3), ('may', 3), ('provide', 2), ('what', 2), ('However', 2), ('time', 2),
('economy', 2), ('or', 2), ('overall', 2), ('level', 2), ('activity', 2), ('people',
2), ('have', 2), ('m', 1), ('sorry', 1), ('but', 1), ('am', 1), ('not', 1), ('able',
1), ('information', 1), ('about', 1), ('current', 1), ('event', 1), ('my', 1), ('kno
wledge', 1), ('limited', 1), ('wa', 1), ('known', 1), ('up', 1), ('until', 1), ('Sep
tember', 1), ('2021', 1), ('general', 1), ('explanation', 1), ('an', 1), ('An', 1),
('period', 1), ('when', 1), ('country', 1), ('slows', 1), ('down', 1), ('contract',
1), ('This', 1), ('happen', 1), ('for', 1), ('many', 1), ('reason', 1), ('such', 1),
('consumer', 1), ('spending', 1), ('investment', 1), ('During', 1), ('harder', 1),
('finding', 1), ('job', 1), ('struggle', 1), ('make', 1), ('profit', 1), ('decline',
1), ('Recessions', 1), ('big', 1), ('impact', 1), ('on', 1), ('s', 1), ('life', 1),
('lead', 1), ('how', 1), ('government', 1), ('operate', 1), ('are', 1), ('usually',
1), ('temporary', 1), ('will', 1), ('eventually', 1), ('start', 1), ('grow', 1), ('a
gain', 1)])
```

Next, we will need to change the key from a word to its frequency rank (e.g., 1st, 2nd, 3rd most frequent word) using a counter.

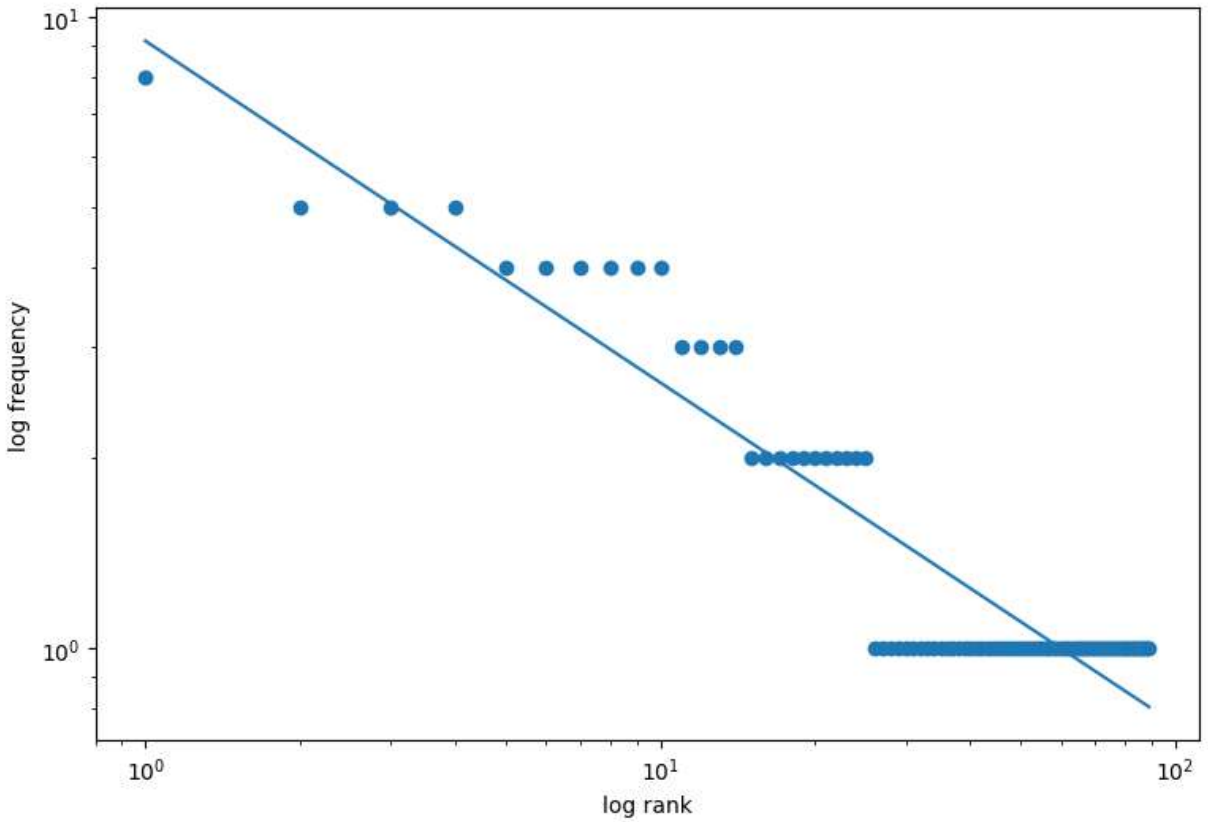
As ordered dictionaries are immutable, we will instead use a new dictionary, final_freq_dict, and add each item in freq_dict to this dictionary.

In [ ]:
```
cntr = 1 #initialise counter
final_freq_dict = collections.OrderedDict() #initliase final frequency dict

for key in freq_dict.keys():
    final_freq_dict[cntr] = freq_dict[key] # add each item to final_freq_dict, and
    cntr += 1 #increment counter

print(final_freq_dict)
```

```
OrderedDict([(1, 8), (2, 5), (3, 5), (4, 5), (5, 4), (6, 4), (7, 4), (8, 4), (9, 4),
(10, 4), (11, 3), (12, 3), (13, 3), (14, 3), (15, 2), (16, 2), (17, 2), (18, 2), (1
9, 2), (20, 2), (21, 2), (22, 2), (23, 2), (24, 2), (25, 2), (26, 1), (27, 1), (28,
1), (29, 1), (30, 1), (31, 1), (32, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1),
(38, 1), (39, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1), (45, 1), (46, 1), (4
7, 1), (48, 1), (49, 1), (50, 1), (51, 1), (52, 1), (53, 1), (54, 1), (55, 1), (56,
1), (57, 1), (58, 1), (59, 1), (60, 1), (61, 1), (62, 1), (63, 1), (64, 1), (65, 1),
(66, 1), (67, 1), (68, 1), (69, 1), (70, 1), (71, 1), (72, 1), (73, 1), (74, 1), (7
5, 1), (76, 1), (77, 1), (78, 1), (79, 1), (80, 1), (81, 1), (82, 1), (83, 1), (84,
1), (85, 1), (86, 1), (87, 1), (88, 1), (89, 1)])
```

Next, we will plot the scatter plot, and its best fit line on a log-log scale.

In [ ]:
```python
x = np.array([int(i) for i in final_freq_dict.keys()]) # frequency rank as x axis
y = np.array([int(i) for i in final_freq_dict.values()]) # frequency percentage as

fig, ax = plt.subplots(figsize = (9, 6))

# change scale to log-log
ax.set_yscale("log")
ax.set_xscale("log")

#set x and y labels
plt.xlabel('log rank')
plt.ylabel('log frequency')

#add plot scatter plot
ax.scatter(x, y)

# calcuate best fit line on a log-log scale (referenced from Craig [2017])
slope, intercept = np.polyfit(np.log(x), np.log(y), 1) # fit log(y) = m*log(x) + c
y_fit = np.exp(slope * np.log(x) + intercept) # calculate the fitted values of y
ax.plot(x, y_fit)

plt.show()
```



In order to calcuate the difference between the text' Zipfian Difference, we will use the absolute difference in the gradient between the plotted line, and the Zipf's law line (which has a gradient of -1 on a log-log scale)

In [ ]:
```
print(f'The line gradient is: {slope}')
print(f'The difference between the line and Zipf\'s law is {np.absolute(-1 - slope)
```

```
The line gradient is: -0.5413753485139352
The difference between the line and Zipf's law is 0.4586246514860648
```

Next, we will gather the above method into a function (minus the line plotting code, as we only need the gradient of the line, and do not need to plot and show the actual graph) called calc_zipfian_diff, so we can call it on each row of the dataframe.

In [ ]:
```
def calc_zipfian_diff(row):
    answer = df_cleaned.at[row, 'lemmatised_answer'] #extract answer
    answer = answer.translate(str.maketrans('', '', string.punctuation)) # strip pu
    tokens = word_tokenize(answer) #tokenise answer
    freq_dist = nltk.FreqDist(tokens) # create freqdist

    freq_dict = dict([(m, n) for m, n in freq_dist.items()]) # create dictionary fr
    freq_dict = collections.OrderedDict(dict(sorted(freq_dict.items(),
                                            key=lambda item: item[1], reverse=True)

    cntr = 1 #initialise counter
    final_freq_dict = collections.OrderedDict() #initliase final frequency dict

    for key in freq_dict.keys():
        final_freq_dict[cntr] = freq_dict[key] # add each item to final_freq_dict,
```

```
        cntr += 1 #increment counter

    x = np.array([int(i) for i in final_freq_dict.keys()]) # frequency rank as x ax
    y = np.array([int(i) for i in final_freq_dict.values()]) # frequency percentage

    # calcuate best fit line on a log-log scale
    slope, intercept = np.polyfit(np.log(x), np.log(y), 1) # fit log(y) = m*log(x)


    zipfian_diff = np.absolute(-1 - slope) # get absolute difference between the gr

    return zipfian_diff
```

Next, we can call the function on each row to find the Zipfian Difference for each answer.

In [ ]:
```
df_cleaned['zipfian_difference'] = 0 #initalise empty column for zipfian difference
```

In [ ]:
```
for i in range(0, len(df_cleaned)):
    df_cleaned.at[i, 'zipfian_difference'] = calc_zipfian_diff(i)
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	Unnamed: 0.1	Unnamed: 0	question	answer	is_human	filtered_answer	lemm
35252	35252	35252	Why is it recommended to pee on someone who ha...	It is n't . Much better to use vinegar , and s...	1	It n't . Much better use vinegar , simply some...	It is n' to us
23770	23770	23770	how long have kanab ambersnail been endangered?	It has been listed as endangered on the United...	1	It listed endangered United States Fish Wildli...	It h enda
40469	40469	40469	Why is n't widescreen a television standard , ...	It was easier to make TVs in the old aspect ra...	1	It easier make TVs old aspect ratio back tv gi...	It wa TVs in

3 rows × 21 columns

2.4.2 Unique Word Count and Stopword Count

We will also extract the unique word proportion and stopword proportion for each text.

We will quantify this by:

1. extracting the proportion of unique words in each answer, divided over the length of the entire answer

2. extracting the proportion of the answer that is made up of stopwords

In [ ]:
```
from nltk import word_tokenize
from nltk.corpus import stopwords
```

In [ ]:
```
# initialise stopwords
stopwords = stopwords.words('english')
```

In [ ]:
```
df_cleaned['unique_word_proportion'] = 0
df_cleaned['stopword_proportion'] = 0
```

In [ ]:
```
for i in range(0, len(df_cleaned)):
    answer = df_cleaned.at[i, 'answer'] #retrieve answer from df
    word_count = df_cleaned.at[i, 'word_count'] #extract word count from df

    tokens = word_tokenize(answer) # tokenise answer

    unique_word_count = len(set(tokens)) # get number of unique words

    tokens_no_stopwords = [token for token in tokens if token not in stopwords] # c
    stopword_count = word_count - len(tokens_no_stopwords) # calculate number of st

    df_cleaned.at[i, 'unique_word_proportion'] = (unique_word_count/word_count) #ca
    df_cleaned.at[i, 'stopword_proportion'] = (stopword_count/word_count) # calcula
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	Unnamed: 0.1	Unnamed: 0	question	answer	is_human	filtered_answer	lemmatised_
6034	6034	6034	Why is voltage used as opposed to current to w...	Voltage is used as a measure of electric hazar...	0	Voltage used measure electric hazard closely r...	Voltage is measure of
40851	40851	40851	What happens when someone gives birth on a pla...	> What citizenship does child hold ? It obviou...	1	> What citizenship child hold ? It obviously v...	> What cit doe child o
42016	42016	42016	the Classes of ships and subs So I 've seen th...	There are many different classes of ships and ...	0	There many different class ship submarine worl...	There a different ship a

3 rows × 23 columns

We will save this to an intermediate csv file

In [ ]:
```
df_cleaned.to_csv('intermediate_data_3.csv', mode='w')
```

2.5 Complex Phrase Features

We will also be extracting the count of 2 different types of complex phrases - Ancient phrases and Cliches.

This will be done by comparing the answer to reference lists of each type of phrase, which have been compiled and cleaned from 2 online sources.

First, we will import the required modules and read in the intermediate csv from the last step

In [ ]:
```
import pandas as pd
import csv
```

In [ ]:
```
df_cleaned = pd.read_csv('intermediate_data_3.csv')
df_cleaned.sample(3)
```

Out[ ]:

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	question	answer	is_human	filtered_answer
312	312	312	312	We have time zones on Earth , so what time is ...	Space does n't have time zones so everyone agr...	1	Space n't time zone everyone agreed say want s...
22220	22220	22220	22220	How do small towns in the US sustain themselve...	Small towns in the US often rely on a mix of l...	0	Small town US often rely mix local business , ...
41757	41757	41757	41757	what is the purpose of education in USA ? Job ...	The purpose of education in the United States ...	0	The purpose education United States help peopl...

3 rows × 24 columns

Next, read in the 2 reference lists from their respective csv files into python lists.

First, we will define a function to convert a csv file to a list

In [ ]:
```
def csv_to_list(filepath):
    # open csv file
    with open(filepath, 'r') as file:
        reader = csv.reader(file) # read csv file
        data = []
        for row in reader:
            data.append(row[0]) # append each row to data list
    return data
```

and call it on the two reference csv files.

In [ ]:
```
cliches = csv_to_list('phrases/cliches.csv')
ancient_phrases = csv_to_list('phrases/ancient_phrases.csv')

print(cliches)
print(ancient_phrases)
```

['ace in the hole', 'ace up your sleeve', 'acid test', 'airing dirty laundry', "all in a day's work", 'all talk', 'all booster', 'all hat', 'all foam', 'all hammer', 'all icing', 'all lime and salt', 'all missile', 'all shot', 'all sizzle', 'all wax and no wick', 'all that and a bag of chips', 'all thumbs', 'all wet', "all's fair in love and war", 'almighty dollar', 'always a bridesmaid', 'ambulance chaser', 'another day', 'ants in your pants', 'apple-pie order', 'arm and a leg', 'armchair quarterback', 'army brat', 'art imitates life', 'artsy-craftsy', 'artsy-fartsy', 'as luck would have it', 'as old as time', 'at loggerheads', 'calm before the storm', 'candle at both ends', "can't cut the mustard", 'case of mistaken identity', 'cat out of the bag', 'cat got your tongue', 'caught red-handed', 'chapter and verse', 'checkered career', 'chickens come home to', 'roost', 'chomping at the bit', 'cleanliness is next to', 'godliness', 'clear as a bell', 'clear as mud', 'cold shoulder', 'E-ticket', 'ear to the ground', 'early bird catches the worm', 'easier said than done', 'easy as 1-2-3', 'easy as pie', 'eat crow', 'eat humble pie', 'enough already', 'even money', 'every dog has its day', 'every fiber of my being', 'everything but the kitchen sink', 'evil twin', 'existential angst', 'experts agree', 'eye for an eye', 'hair of the dog', 'hard to believe', 'have a nice day', 'head honcho', "heart's content", 'hell-bent for leather', 'hidden agenda', 'high and the mighty', 'high on the hog', 'hold a candle to', 'hold your horses', 'hold your tongue', 'hook or by crook', 'horse of a different color', 'hot knife through butter', 'how goes the battle?', 'keep an eye on you', 'keep it down', 'keep it simple', 'keep up with the Joneses', 'keep your cards close to vest', 'keep your chin up', 'keep your fingers crossed', 'keep your powder dry', 'kick ass', 'kickbutt', 'kick the bucket', 'kick up your heels', 'kick you to the curb', 'kick your feet up', 'kid in a candy store', 'kill two birds with one stone', "life's a bitch", 'lighten up', 'lights out', 'like a sore thumb', 'like butter', 'like the plague', "like there's no tomorrow", "lion's share", 'litmus test', 'little black book', 'live and learn', 'long and short of it', 'long lost love', 'look before you leap', 'lounge lizard', 'loved and lost', 'low man on the totem pole', 'luck of the draw', 'luck of the Irish', 'off the wagon', 'old college try', 'old meets new', 'older and wiser', 'older than dirt', 'older than Methuselah', 'on the bandwagon', 'on the nose', 'on the wagon', 'on thin ice', 'one born every minute', 'one foot in the grave', 'one in a million', 'only game in town', 'only to be met', 'out of pocket', 'out of the frying pan', 'out on a limb', 'safe than sorry', 'salt of the earth', 'save face', 'scared stiff', 'scared to death', "school's out", 'screaming meemies', 'senses reel', 'set the record straight', 'shake a stick should of', 'shoulder to the wheel', 'shouldered his way', 'shut your trap', 'sigh of relief', 'significant other', 'silence is golden', 'take one for the team', 'take the bull by the horns', 'take the plunge', 'takes one to know one', 'talk turkey', 'ten foot pole', 'the earth moved', 'the final analysis', 'the real McCoy', 'the same old story', 'these things happen', 'thick as thieves', 'think outside of the box', "third time's the charm", 'under the gun', 'under the same roof', 'understated elegance', 'unexpected twist', 'until the cows come home', 'up his sleeve', 'up the creek', 'up the wrong tree', 'very real concern', 'view with alarm', 'babe in the woods', 'back against the wall', 'back in the saddle', 'back to square one', 'back to the drawing board', 'bad to the bone', 'badge of honor', 'Badonkadonk', 'ballpark figure', 'balls to the wall', 'baptism of fire', 'bare bones', 'bark is worse than the bite', 'bark up the wrong tree', 'bat out of hell', 'bats in the belfry', 'battle royal', 'beat around the bush', 'beat the bushes', 'beats me', 'behind the eight ball', 'bent out of shape', 'best foot forward', 'bet your bottom dollar', 'better half', 'better late than never', 'better mousetrap', 'better safe than sorry', 'better than ever', 'better the Devil you know', 'between a rock and a hard place', 'beyond the pale', 'bib and tucker', 'big as life', 'big fish in a small pond', 'big man on campus', 'bigger they are', 'communist conspiracy', 'conniption fit', 'could care less', "couldn't care less", "couldn't get to first base", 'count your blessings', 'countless hours', 'creature comfort', 'crime in the street', 'curiosity killed the cat', 'curry favor', 'cut a fine figure', 'cut and dried', 'cut to the chase', 'cut to the quick', 'cute as a

button', 'facts of life', 'fair-haired one', 'fair weather friend', 'fall off of a t
urnip truck', 'fat slob', 'favor us with a song', 'fear and loathing', 'feather your
nest', 'fellow traveler', 'few and far between', 'field this one', 'fifteen minutes
of fame', 'fish nor fowl', 'fly by night', 'fly the coop', 'for the birds', 'fox in
the henhouse', 'freudian slip', 'fun and games', 'fun in the sun', 'I beg to diffe
r', 'if the shoe fits', "I'm okay", 'in a nutshell', 'in a pinch', 'in a wink', "in
harm's way", 'in the tank', 'in your dreams', 'in your face', 'inexorably drawn', 'i
nfo dump', 'influence peddling', 'intents and purposes', 'it was a dark and stormy n
ight', "it won't fly", "King's English", "king's ransom", 'kiss and tell', 'kiss as
s', 'kiss of death', 'kit and kaboodle', 'knee-high to a grasshopper', 'knock it out
of the park', 'knock on wood', 'knock your socks off', 'knocked up', 'know him from
Adam', 'know the ropes', 'know the score', 'knuckle down', 'knuckle sandwich', 'knuc
kle under', 'make my day', 'male chauvinist', "man's best friend", 'many moons', 'ma
ny-splendored thing', 'mark my words', 'meaningful relationship', 'mellow out', 'mom
ent of glory', "moment's respite", 'Monday morning quarterback', 'monkey suit', 'mon
key see', 'motherhood and apple pie', 'movers and shakers', 'moving experience', 'my
two cents', "p's and q's", 'pain and suffering', 'panic button', 'party pooper', 'pa
tter of little feet', 'pass the sniff test', 'pay through the nose', 'peas in a po
d', 'perfect storm', 'pig in a poke', 'pillar of society', 'pinko', 'plenty of fish
in the sea', 'poison pen', 'poor as a churchmouse', 'poor excuse for', 'pot calling
the kettle black', 'proud possessor', 'put my foot down', 'put your foot down', 'qui
ck as a bunny', 'quick and the dead', 'slept like a log', 'small world', 'snake in t
he grass', 'snow job', 'snug as a bug', 'some of my best friends', 'something the ca
t dragged in', 'spade a spade', 'spare the rod', 'spitting image', 'spring to life',
'this day and age', 'this point in time', "three strikes and you're out", 'through t
he grapevine', 'throw in the towel', 'tiger by the tail', 'till the fat lady sings',
'time and time again', 'time is of the essense', 'tip of the iceberg', "'tis the sea
son", 'to err is human', 'to the best of my knowledge', 'wakeup call', 'was my face
red', 'watch your tongue', 'web of intrigue', 'week of sundays', 'what a bummer', 'w
hat comes around', 'what the cat dragged in', 'what the dickens', 'what the heck',
'what the hell', 'what you see is what you get', "what's not to like", 'wheeler-deal
er', 'when in doubt', 'when push comes to shove', 'when rubber meets the road', "whe
n the cat's away", 'when the going gets tough', 'bird in the hand', 'birds and the b
ees', 'birds of the feather', 'bite the dust', 'bite your tongue', 'bitter disappoin
tment', 'black as coal', 'blast from the past', 'bleeding heart', 'blind as a bat',
'blood is thicker than water', 'blood money', 'blood on your hands', 'blood sweat an
d tears', 'blow this pop stand', 'blow this joint', 'blushing bride', 'boil it down
to', 'bone of contention', 'booze and broads', 'bored to tears', 'born and raised',
'born with a silver spoon in your mouth', 'born yesterday', 'bottom line', 'brain dr
ain', 'brain dump', 'brass tacks', 'bring home the bacon', 'broken record', "brothe
r's keeper", 'bull by the horns', 'bull in a china shop', 'bump in the night', 'busy
as a bee', 'but seriously', 'by and large', 'darkest before the dawn', 'dead as a do
ornail', 'death and destruction', 'death and taxes', "death's doorstep", 'devil is i
n the details', 'dim view', 'dog days', 'dog in the manger', "don't count your chick
ens before they're hatched", "don't do the crime if you can't do the time", 'doubtin
g Thomas', 'down and dirty', 'down in the dumps', 'down pat', 'down the toilet', 'do
wn the drain', 'down the hatch', 'down to earth', 'drive you up a wall', 'dutch uncl
e', 'dyed in the wool', 'garbage in', 'get the sack', 'get your groove back', 'gets
my goat', 'gift horse in the mouth', 'gilding the lily', 'give a damn', 'give me a b
reak', 'gives me the creeps', 'go him one better', 'goes without saying', 'good deed
for the day', 'good time was had by all', 'greek to me', 'green thumb', 'green-eyed
monster', 'grist for the mill', 'guiding light', 'Jack of all trades', 'jockey for p
osition', 'Johnny-come-lately', 'joined at the hip', 'jump down your throat', 'jump
in with both feet', 'jump on the bandwagon', 'jump the gun', 'jump his bones', 'jump
her bones', 'junk in the trunk', 'jury is still out', 'justice is blind', 'labor of
love', 'lap of luxury', 'last but not least', 'last-ditch effort', 'last hurrah', 'l

aw of the jungle', 'law of the land', 'lay down the law', 'leaps and bounds', 'let s
leeping dogs lie', 'let the cat out of the bag', "let's split", 'liberal media', 'li
e like a rug', 'life and limb', 'life imitates art', 'neat as a pin', 'needless to s
ay', 'nip it in the bud', 'no guts', 'no love lost', 'no pain', 'no stone unturned',
'no time like the present', 'nose to the grindstone', 'not in my back yard', 'not on
your tintype', 'number one fan', 'numerous to mention', 'radical chic', 'rags to ric
hes', 'raining buckets', 'raining cats and dogs', 'rank and file', 'read my lips',
'red herring', 'redheaded stepchild', 'reign supreme', 'remember the alamo', 'road t
o hell is paved with good intentions', 'rob Peter to pay Paul', 'rock and a hard pla
ce', 'rocket scientist', 'rocket science', 'rope a dope', 'run it up the flagpole',
'running dog lackey', 'squeaky wheel gets the oil', 'squeaky wheel gets the grease',
'start from scratch', 'stick in the mud', 'stick in your craw', 'still waters run de
ep', 'stop and smell the roses', 'store bought', 'stranger than fiction', "straw tha
t broke the camel's back", 'stubborn as a mule', 'stuff that dreams are made of', 's
tuffed shirt', 'tongue-in-cheek', 'too hot to handle', 'touch of blarney', 'tough as
nails', 'tough luck', 'tough row to hoe', 'traditional family values', 'trials and t
ribulations', 'tried and true', 'trip down memory lane', 'true blue', 'turn your smi
le upside-down', 'turn your frown upside-down', 'twist of fate', 'twists and turns',
'two to tango', 'who has everything', 'whole ball of wax', 'whole hog', 'whole nine
yards', 'whole other story', 'wild goose chase', 'wild oats', 'will wonders never ce
ase', 'wimp', 'win friends and influence', 'people', 'win one for the Gipper', 'wnni
ng is everything', 'wisdom of the ages', 'without benefit of clergy', 'wolf at the d
oor', 'words fail', 'work like a dog', 'worst nightmare', 'wrong side of the bed']
['abodement', 'abram', 'abroach', 'abrook', 'absey-book', 'aby', 'accdant', 'accit
e', 'accomplement', 'accompt', 'accouter', 'accoutre', 'Acheron', 'acknow', 'aconitu
m', 'acquittance', 'acture', 'Actus Primus', 'Actus Quartus', 'Actus Quintus', 'Actu
s Secundus', 'Actus Tertius', 'addle', 'adieu', 'adoptious', 'advocation', 'aery',
'afeared', 'affiance', 'affined', 'affy', 'agate', 'agazed', 'aglet', 'agood', 'agu
e', 'aiery', 'alack', 'alarum', 'Albion', 'alderliefest', 'alder-liefest', 'alder-li
evest', 'alewife', 'ale-wife', 'allayment', 'allegiant', 'allottery', 'Almain', 'alm
s drink', 'almsdrink', 'almshouse', 'amain', 'amerce', 'ames-ace', 'amort', 'anatomi
ze', 'ancient', 'a-night', 'annexment', 'anon', 'antagonist', 'antechamber', 'Anthro
pophagi', 'Anthropophaginian', 'antic', 'antick', 'anticked', 'antre', 'appeach', 'a
ppellant', 'apperil', 'apple-john', 'approof', 'apricock', 'aqua vitae', 'aqua-vita
e', 'argal', 'Argier', 'argosy', 'aright', 'armigero', 'aroint', 'a-row', 'arrant',
'arras', 'aside', 'asinego', 'assay', 'assemblance', 'assinego', 'assubjugate', 'ato
mies', 'attaint', 'attainture', 'atwain', 'a-twain', 'audit', 'auld', 'avaunt', 'avo
uch', 'aye', 'bacare', 'baccare', 'bacchanal', 'backare', 'baffle', 'baggage', 'Baja
zet', 'balcony in a Shakespearean theater', 'ballow', 'balsamum', 'ban', 'bandog',
'ban-dog', 'banns', 'Barbary', 'Barbason', 'barbermonger', 'barber-monger', 'barm',
'barnes', 'barricado', 'basilisk', 'basta', 'bastard', 'bastardy', 'bastinado', 'bat
e', 'bat-fowling', 'batler', 'batlet', 'battaille', 'battalia', 'bavin', 'bawbling',
'bawcock', 'bawdry', 'beadsman', 'beam', 'bearing cloth', 'bearing-cloth', 'beaver',
'bed of Ware', 'bed-hangings', 'bedlam', 'beetle', 'befall', 'begirt', 'behoveful',
'beldam', 'belike', 'bemadding', 'bemete', 'be-mete', 'bemoil', 'be-moil', 'be-nette
d', 'benison', 'Bergomask', 'bescreened', 'be-screened', 'beseech', 'beshrew', 'besl
ubber', 'besort', 'bespake', 'bespeak', 'bespoke', 'bestraught', 'betake', 'beteem',
'betid', 'betide', 'betimes', 'betossed', 'betwixt', 'bewray', 'bezonian', 'biding',
'biggen', 'bilberry', 'bilbo', 'bill', 'birdbolt', 'bird-bolt', 'bisson', 'blench',
'bob', 'bodement', 'bodge', 'Bodkin', 'boÓtier vert', 'bolster', 'bolted', 'bolter',
'bolting hutch', 'bolting-hutch', 'bombard', 'bona roba', 'bona-roba', 'bondman', 'b
onny', 'boot', 'bootless', 'bosky', 'botcher', 'bots', 'bottled spider', 'bounden',
'bourn', 'brabble', 'brach', 'brainish', 'brainpan', 'brain-pan', 'brake', 'braver
y', 'breese', 'Bretagne', 'brinded', 'brinish', 'Britaine', 'Brittany', 'brock', 'br
ogue', 'bruit', 'bubukle', 'buckler', 'buckram', 'bum', 'bung', 'burden as a music t
erm', 'burgonet', 'burthen', 'buskin', 'busky', 'buss', 'buzzer', 'by-room', 'cacoda

emon', 'cacodemon', 'caddis', 'caitiff', 'Calipolis', 'caliver', 'callat', 'callet', 'Cam', 'cambric', 'camlet', 'canakin', 'canary', 'canis', 'cannikin', 'canzonet', 'cap-a-pie', 'capon', 'carbuncle', 'carcanet', 'cardmaker', 'carlot', 'carract', 'cart', 'carviare', 'casing', 'casque', 'cataplasm', 'catch', 'catchword', 'cater', 'cater-cousin', 'caterwauling', 'cautel', 'cautelous', 'caviary', 'cavil', 'cellarage', 'cellar-age', 'century', 'cerecloth', 'cerement', 'certes', 'chace', 'champain', 'chanticleer', 'chantry', 'chape', 'chapfallen', "chap-fall'n", 'chapless', 'chapman', 'chapt', 'charactery', 'charneco', 'charnel house', 'charnel-house', 'cheveril', 'chid', 'chine', 'chirurgeon', 'chopine', 'chorus', 'chough', 'chuck', 'cicatrice', 'cinque-pace', 'Cinque-ports', 'circummure', 'civet', 'clackdish', 'clack-dish', 'clepe', 'clept', 'clew', 'climature', 'clinquant', 'clog', 'cloistress', 'clotpoll', 'clout', 'clouted brogues', 'clovest', 'cloyless', 'cockatrice', "cockerel's stone", 'cockle', 'cockle hat', 'cockleshell', 'cockney', 'cock-shut time', 'codpiece', 'coffer', 'cog', 'cogging', 'coign', 'coil', 'collop', 'combinate', 'complice', 'complot', 'compulsative', 'con', 'concernancy', 'concupy', 'condign', 'coney', 'confixed', "conge'd", 'congeed', 'conger', 'congied', 'conjunctive', 'conjurations', 'conserve', 'consort', 'conspectuity', "constring'd", 'constringed', 'contemn', 'continuate', 'contumeliously', 'conventicle', 'convive', 'cony', "copp'd", 'copped', 'coppice', 'coram', 'coranto', 'coronal', 'corse', 'corselet', 'costard', 'costermonger', 'cotquean', 'coulter', 'couplement', 'covert', 'coy', 'coystril', 'coystrill', 'coz', 'cozen', 'cozenage', 'cozier', 'crare', 'crescive', 'crotchets', 'crown', 'crownet', 'crupper', 'crusado', 'cruzado', 'cubiculo', 'cuckold', 'cuckoo flower', 'cullion', 'dace', 'daff', 'dam', 'damosella', 'damson', 'Danskers', 'darnel', 'darraign', 'dauberry', 'dauphin', 'debile', 'debosh', 'defeature', 'deflower', 'defunct', 'delated', 'delation', 'demi-wolf', 'denay', 'denier', 'denotement', 'depute', 'descant', 'designments', 'despatch', 'despite', 'determinate', 'diadem', 'dialogue', 'diaper', 'dich', 'dirge', 'disannul', 'disme', 'dispark', 'dispatch', 'disprized', 'disproperty', 'disquantity', 'dissentious', 'distaff', 'distained', 'distemperature', 'distract', 'distrain', 'ditch-dog', 'dive-dapper', 'divers', 'doit', 'dole', 'dormouse', 'dotant', 'doublet', 'dout', 'dower', 'dowlas', 'dowle', 'down-gyved', 'Downs', 'drab', 'drachma', 'draff', 'dram', 'dramatis personae', 'drave', 'drawer', 'drossy', 'drumble', 'dry-beat', 'dryfoot', 'dudgeon', 'duello', 'dug', 'dump', 'dup', 'durance', 'durst', 'eale', 'eaning', 'eftest', 'eftsoons', 'eisel', 'eld', 'embarquement', 'embassage', 'emblaze', 'embossed', 'embowel', 'embracement', 'empiricutic', 'emulous', 'encave', 'enchase', 'encloud', 'encompassment', 'enfeoff', 'engirt', 'englut', 'enmew', 'enow', 'enrank', 'enroot', 'enround', 'entail', 'enter', 'enwombed', 'epilogue', 'epitheton', 'equipage', 'ere', 'erewhile', 'errant', 'erst', 'escot', 'esperance', 'espial', 'essay', 'estridge', 'eterne', 'Ethiop', 'Ethiope', 'ever yet', 'everyet', 'exchequer', 'excursion', 'exeunt', 'exion', 'exit', 'exposture', 'exsufflicate', 'extirp', 'extraught', 'eyas', 'eyne', 'eyrie', 'facinorous', 'factionary', 'fadge', 'fain', 'fait', 'falchion', 'fallow', 'falsing', 'fap', 'farced', 'fardel', 'farthing', 'fatigate', 'fay', 'fealty', 'feat', 'fecks', 'federary', "fee'd", 'feeder', 'fee-grief', 'Fee-simple', 'felly', 'feodary', 'fere', 'festinately', 'fico', 'fie', 'fillip', 'filly foal', 'fine sovereign', 'firago', 'firk', 'fitchew', 'flapdragon', 'fleer', 'fleshment', 'flews', 'flexure', 'flirt-gill', 'flock', 'fob', 'fob off', 'foh', 'foil', 'foin', 'foison', 'fool', 'foot', 'footboy', 'foot-cloth mule', 'footing', 'fordo', 'forsooth', 'fortnight', 'fosset', 'foughten', 'foul papers', 'foxship', 'fracted', 'frank', 'fraught', 'fraughtage', 'frippery', 'frontlet', 'froward', 'frush', 'fullam', 'fumiter', 'fumitory', 'fust', 'fustilarian', 'gaberdine', 'gad', 'gage', 'gainsay', 'gallery', 'galliard', 'galliass', 'galloglasses', 'gallow', 'gallowglasses', 'gamesome', 'gammon', 'gaoler', 'garboil', 'garner', 'gaskins', 'gasted', 'gawd', 'geck', 'gentle', 'gest', 'gib', 'giglot', 'gimmal', 'gimmer', 'gimmor', 'ging', 'gird', 'Gis', 'glaive', 'glanders', 'glave', 'gleek', 'glister', 'glose', 'glow-worm', 'gloze', 'go to the world', 'gobbet', 'goblin', 'godwit', 'good den', 'good-year', 'gorbellied', 'gord', 'gourd', 'gramercy', 'gratis', 'graymalkin', 'greave', 'greenly', 'gripe', 'grise', 'grize', 'groat', 'guardant', 'gudgeon', 'gue

rdon', 'gules', 'gull', 'gyve', 'habiliment', 'habitude', 'haggard', 'hale', 'half-c
rown', 'halfpenny', 'halidom', 'halter', 'handsaw', 'hap', "ha'penny", 'hapless', 'h
aply', 'hard-handed', 'hardiment', 'harebell', 'hare-bell', 'harlock', 'harrow', 'ha
sle', 'haste-post-haste', 'hatchment', 'haught', 'hautbois', 'hautboy', 'haviour',
'head', 'head-lugged', 'hebona', 'hectic', 'hedge', 'hedge-pig', 'hedgepriest', 'hef
t', 'hefted', 'heigh', 'helm', 'hempen', 'henchman', 'hent', 'hest', 'hewgh', 'Hiem
s', 'high cross', 'high-cross', 'hight', 'hilding', 'hindmost', 'hither', 'hizz', 'h
odge pudding', 'hodge-pudding', 'hoise', 'holidame', 'holla', 'hollaing', 'holp', 'h
oly-ales', 'honey-stalks', 'honorificabilitudinitatibus', 'hornbook', 'horn-mad', 'h
orologe', 'hothouse', 'hot-house', 'house of resort', 'howbeit', 'hox', 'hoy', 'hugg
ermugger', 'hull', 'hurly', 'hurricano', 'hurtle', 'husband', 'huswife', 'hyen', 'Hy
rcanian beast', 'I wis', 'ignomy', "'ild", 'illume', 'imbar', 'imbare', 'imbrue', 'i
mmanity', 'immoment', 'imp', 'impartment', 'impasted', 'impawn', 'imperceiverant',
'impeticos', 'impleached', 'implorator', 'impone', 'importunacy', 'imposthume', 'imp
rese', 'impressure', 'imputation', 'incarnadine', 'incensement', 'incertain', 'inchm
eal', 'inch-meal', 'inclip', 'incony', 'incorpsed', 'increase', 'indent', 'indiffere
ncy', 'indigest', 'indign', 'indue', 'indurance', 'infamonize', 'inferreth', 'infort
unate', 'inhearse', 'Inhoop', 'injointed', 'inkhorn', 'inkle', 'inscroll', "inscul
p'd", 'insculped', 'insculpted', 'insculpture', "insinew'd", 'insinewed', 'insistur
e', "insteep'd", 'insteeped', 'intelligencer', 'intendment', 'intenible', 'intrins
e', 'inventorially', 'iron crow', 'irregulous', 'issue', 'iwis', 'jack-a-lent', 'jac
ks', 'jade', 'jakes', 'jar', 'jaunce', 'jerkin', 'jester', 'jet', 'jocund', 'John-a-
dreams', 'joint-ring', 'jordan', 'jot', 'jutty', 'juvenal', 'kam', 'kecksy', 'keec
h', 'keel', 'Keisar', 'ken', 'keptst', 'kern', 'kersey', 'kibe', 'kicky-wicky', 'kin
e', 'kirtle', 'knack', 'knap', 'knight-errant', 'la', 'labras', 'lakin', 'Lammas',
'Lammas-tide', 'lampass', 'lank', 'lanthorn', 'lapwing', 'lass-lorn', 'lated', 'latt
en', 'laund', 'lazar', 'leagued', 'leather coat', 'leather-coat', 'leathern', 'leave
s', 'lees', 'leet', 'legerity', 'leiger', 'leman', "l'envoy", 'levy', 'libbard', 'li
ct', 'lief', 'liege', 'liegeman', 'lifeling', 'limbeck', 'Limbo', 'lime', 'lineal',
'lineament', 'linstock', 'list', 'lists', 'lither', 'loach', 'loaden', 'lob', 'lockr
am', 'lodestar', 'lode-star', 'loggat', 'loggats', 'loof', 'lour', 'lout', 'lown',
'lowt', 'lozel', 'lubber', 'luce', 'lune', 'lustihood', 'lym', 'maggot-pie', 'magnif
ico', 'Mahu', 'main', 'mained', 'makeless', 'malapert', 'malkin', 'malmsey', 'malt w
orm', 'mammer', 'mammet', 'mammock', 'manakin', 'mandragora', 'manikin', 'Manningtre
e', 'marchpane', 'marish', 'marl', 'Marry', 'mart', 'martinmas', 'martlemas', 'martl
et', 'Mary-buds', 'masque', 'matin', 'maugre', 'maw', 'mazzard', 'meacock', 'measle
s', 'medlar', 'meed', 'meetest', 'meetness', 'mehercle', 'meiny', 'mell', 'mercatant
e', 'mered', 'mete-yard', 'metheglin', 'methinks', 'mew', 'mickle', 'milch', 'milkpa
ps', 'minikin', 'minim', 'minimus', 'mirable', 'miscarry', 'misconster', 'misprise',
'misproud', 'missingly', 'mis-tempered', 'mobled', 'Modo', 'moe', 'moiety', 'moldwar
p', 'mome', 'momentany', 'monstruosity', 'montant', 'mooncalf', 'Morisco', 'morris',
'morris-pike', 'morrow', 'mote', 'mow', 'mummy', 'muniment', 'mure', 'murrain', 'mur
ther', 'musk rose', 'musk-rose', 'mutine', 'nayward', 'nayword', 'neaf', 'near-legge
d', 'neat', 'needly', 'neeld', 'neeze', 'neif', 'netherstocks', 'nether-stocks', 'no
ddle', 'nonce', 'nose', "not'st", 'nousle', 'novum', 'nowl', 'noyance', 'nuncle', 'n
uthook', 'obsequies', 'oeilliade', 'oeilliades', "o'ercrow", "o'erleaven", "o'er-lea
ven", "o'er-raught", "o'er-read", "o'erskip", "o'ertrip", 'oneyer', 'ope', 'oppugnan
cy', 'orange-wife', 'ordinant', 'orgulous', 'orison', 'orthographer', 'ostler', 'oup
h', 'ouphe', 'ousel-cock', 'outface', 'outsport', 'outwall', 'out-wall', 'overborn
e', 'overcrow', 'overglance', 'overleather', 'overleaven', 'over-raught', 'overrea
d', 'overskip', 'overtrip', 'overweening', 'over-weening', 'paction', 'paddock', 'pa
jock', 'pale', 'pall', 'palliament', 'palmy', 'palter', 'paly', 'pantler', 'panyn',
'panynæ', 'pap', 'paraquito', 'parcel', 'pardonnez-moi', "'parit", 'parle', 'parlin
g', 'parmaceti', 'Parthia', 'partisan', 'partizan', 'pash', 'passado', 'passymeasur
e', 'passy-measure', 'pastern', 'patch', 'patine', 'pauca', 'pavan', 'pavane', 'pavi
n', 'peat', 'pedascule', 'peise', 'pell-mell', 'penny', 'pennyworth', 'penny-worth',

'peradventure', 'perdie', 'perdona-mi', 'perdu', 'perdurable', 'perdy', 'periapt',
'perpend', 'persever', 'perspective', 'pestiferous', 'petar', 'petard', 'pettish',
'pew-fellow', 'pheeze', 'Phoebe', 'physic', 'Picht-hatch', 'pickthank', 'pick-than
k', 'Pickt-hatch', 'pight', 'pilchard', 'pill', 'pinfold', 'pinnace', "pin's fee",
'pioned', 'pioner', 'pippin', 'pish', 'pismire', 'pittikin', 'placket', 'plain', 'pl
aining', 'planched', 'plats', 'plausive', 'pleached', 'pleb', 'plebeii', 'plurisy',
'poesy', 'point-device', 'poise', 'poke', 'politic', 'politicly', 'poll', 'polled',
'pomewater', 'poniard', 'pontifical', 'poor John', 'poperin', 'popinjay', 'porpentin
e', 'porringer', 'portance', 'posset', 'post horse', 'postern', 'post-horse', 'potat
ion', 'potch', 'potting', 'pottle', 'poulter', 'pouncet box', 'pouncet-box', 'poun
d', 'power', 'practisant', 'pratest', "prat'st", 'preceptial', 'prejudicate', 'preno
minate', 'prentice', "'prentice", 'cardecu', 'coif', 'quait', "quart d'?cu", 'quat',
'quatch', 'quean', 'quern', 'questant', 'questrist', 'quiddit', 'quillet', 'quintai
n', 'quire', 'quittance', 'quiver', 'quoif', 'quoits', 'quondam', 'quotha', 'quoth-
a', 'quotidian', 'rabato', 'rabbit-sucker', 'rack', 'rampallian', 'raught', 'ravin',
'rearmouse', 'rear-mouse', 'reave', 'rebato', 'recheat', 'reck', 'recordation', 'rec
reant', 'red plague', 'rede', 'reechy', 'reel', 'refel', 'reft', 'reguerdon', 'rejou
rn', 'relique', 'remember', 'remembrancer', 'remotion', 'repair', 'repugn', 'reputel
ess', 'reremouse', 'rere-mouse', 'reverb', 'Rhenish', 'rheum', 'rial', 'rib', 'riban
d', 'riggish', 'rigol', 'rivality', 'robustious', 'roe', 'roist', 'romage', 'ronyo
n', 'rood', 'rooky', 'ropery', 'roundure', 'rowel', 'royal', 'roynish', 'rub', 'rudd
ock', 'rudesby', 'ruff', 'runagate', 'rushling', 'ruth', 'ryal', 'sackbut', 'sacring
bell', 'sain', "Saint Colme's Inch", 'saith', 'Sala', 'Salique Law', 'sallet', 'sanc
tuarize', 'sans', 'savour', 'saw', 'scab', 'scaffoldage', 'scald', 'scambling', 'sca
mel', 'scant', 'scantling', 'scape', "'scape", 'scarf up', 'scarfed', 'scathful', 's
conce', 'scotch', 'scrimer', 'scrip', 'scrowl', 'scroyle', 'scull', 'scullion', 'scu
rril', 'scutch', 'sea gown', 'sea-gown', 'sea-mark', 'seel', 'self', 'semblable', 's
emblative', 'sennet', 'senseless-obstinate', 'sequent', 'sequester', 'serpigo', 'ses
sa', 'Setebos', 'setter', 'several', 'shale', 'shards', "shark'd", 'sharked', 'sheal
ed', 'shelvy', 'shent', "sheriff's post", 'shilling', 'shive', 'shoon', 'shotten',
'shough', 'shovel board', 'shovel-board', 'shrieve', 'shrift', 'shrive', 'shunless',
'sicle', 'signiory', 'signory', 'simples', 'simular', 'sirrah', 'sith', 'sithence',
'sixpence', 'skainsmates', 'skains-mates', 'skeins-mates', 'skimble-skamble', 'skink
er', 'skirr', 'sleided', 'sleight', 'sliding', 'slips', 'slops', 'slovenry', 'slutti
sh', 'smatch', 'smilets', 'smit', 'smooth', 'smoothing', 'Smulkin', 'smutch', 'snea
p', 'sneaping', 'sneck up', 'snuff', 'soft', 'soilure', 'solidare', 'sonties', 'soot
h', 'sortance', 'souse', 'sowl', 'sowle', 'spavin', 'speken', 'sperr', 'spilth', 'sp
irt', 'spital', 'splenitive', 'spongy', 'springe', 'springhalt', 'squier', 'squiny',
'stale', 'startingly', 'statist', 'statua', 'staves', 'stell', 'stile', 'stillitor
y', 'stilly', 'stoccado', 'stomacher', 'stonebow', 'stone-bow', 'stoup', 'strappad
o', 'straw', 'strond', 'strucken', 'strumpet', 'subscription', 'succour', 'sumpter',
'superflux', 'supervise', 'suppliance', 'supportance', 'supposal', 'sur-', 'suranc
e', 'surplice', 'suspire', 'suum cuique', 'swag', 'swain', 'swashing', 'sweeting',
'Switzer', 'swound', 'tabor', 'tabour', 'tainture', 'tallow', 'tang', 'tanned', 'tar
ge', 'Tarpeian', 'tarre', 'tarry', 'teem', 'tench', 'tendance', 'tent', 'Termagant',
'teston', 'testoon', 'tetter', 'Tewkesbury mustard', 'thane', 'tharborough', 'Thasso
s', 'theorick', 'thereunto', 'therewithal', 'thews', 'thine', 'thither', 'thorough',
'thou', "thou'dst", 'thraldom', 'thrall', 'thrasonical', 'three-man beetle', 'threep
ence', 'threne', 'thrum', 'thrummed hat', 'thy', 'thy', 'thyself', 'tilt', 'tilter',
'tiltyard', 'tilt-yard', 'tirrit', 'tortive', 'touse', 'toy', 'traject', 'trencher',
'trollmydames', 'troll-my-dames', 'troth', 'trow', 'Troyan', 'truckle-bed', 'truepen
ny', 'trull', 'truncheon', 'trundle-tail', 'tuck', 'tucket', 'tun', 'tundish', 'tu
p', 'Turlygood', 'twangling', 'twelvemonth', 'twiggen', 'twilled', 'twink', 'twire',
'twit', 'twopence', "unanel'd", 'unaneled', 'unbarbed', 'unbated', 'unbodied', 'unbo
lted', 'unbreathed', 'uncape', 'unclew', 'uncoined', 'uncurrent', 'underfiends', 'un
derskinker', 'under-wrought', 'uneath', 'unfeigned', 'ungenitured', 'ungravely', 'un

```
hatched', "unhousel'd", 'unhouseled', 'unlineal', 'unmanned', 'unmuzzle', 'unparagon
ed', 'unpossessing', 'unprizable', 'unprizeable', 'unproperly', 'unseminared', "unse
minar'd", 'unsifted', 'unsinewed', "unsinew'd", 'unsisting', 'unstanched', 'unstaunc
hed', 'untemper', 'untempering', 'unthrift', 'upspring reel', 'usance', 'utis', 'vad
e', 'vail', 'vantbrace', 'varlet', 'vassalage', 'vastidity', 'vault', 'vault', 'vaul
t', 'vault', 'vaultage', 'vaulty', 'vaunt-courier', 'vaward', 'vegetives', 'velure',
'vendible', 'venew', 'veney', 'ventage', 'verger', 'verily', 'vesper', 'vesture', 'v
ia', 'viands', 'victuals', 'videlicet', 'viewless', 'villany', 'vinewedst', 'vinnewe
d', 'vinewed', 'virginalling', 'vizament', 'vizard', 'votary', 'vouchsafe', 'waftag
e', 'wafture', 'waggoner', 'wailful', 'wainropes', 'wain ropes', 'wall-eyed', 'wall-
newt', 'Walloon', 'wanion', 'wanned', 'wappened', 'wannion', "wann'd", "wappen'd",
'warrantise', 'water-galls', 'water-rug', 'water-work', 'weal', 'wealsman', 'weeds',
'weet', 'welkin', 'Welsh hook', 'westward-ho!', 'wezand', 'wesand', 'weasand', 'whea
ten', 'Wheeson', 'whelk', 'whence', "wh'er", "whe'r", 'wherefore', 'wheyface', 'whey
-face', 'whiffler', 'while-ere', 'whiles', 'whimple', 'whipstock', 'whist', 'whitin
g', 'whitster', 'whittle', 'whoo-bub', 'whoobub', 'wight', 'willow garland', 'window
-bars', 'windring', 'winter-ground', 'wis', 'wistly', 'withal', 'wittol', 'womby',
'wont', 'wont', 'wonted', 'woodbine', 'woodcock', 'woolward', 'wot', 'wrack', 'wrack
ful', 'writhled', 'wroth', 'yare', 'y-clad', 'yclept', 'yclad', 'ycleped', 'yellow
s', 'yeoman', 'yeoman', 'yeoman', 'yerk', 'yesty', 'yond', 'younker', 'zounds']
```

Next, we will also need to lemmatise all the phrases in each phrase list for comparison (as the answers will also be lemmatised)

First, we will define a function to lemmatise all phrases in a phrase list

In [ ]:
```python
def lemmatise_phrase_list(phrase_list):
    output_list = []
    lemmatizer = WordNetLemmatizer()
    for phrase in phrase_list:
        tokens = word_tokenize(phrase)
        lemmatised_tokens = [lemmatizer.lemmatize(token) for token in tokens]
        output_list.append(' '.join(lemmatised_tokens))

    return output_list
```

In [ ]:
```python
cliches_lemmatised = lemmatise_phrase_list(cliches)
ancient_lemmatised = lemmatise_phrase_list(ancient_phrases)

print(cliches_lemmatised)
print(ancient_lemmatised)
```

['ace in the hole', 'ace up your sleeve', 'acid test', 'airing dirty laundry', "all in a day 's work", 'all talk', 'all booster', 'all hat', 'all foam', 'all hammer', 'all icing', 'all lime and salt', 'all missile', 'all shot', 'all sizzle', 'all wax and no wick', 'all that and a bag of chip', 'all thumb', 'all wet', "all 's fair in love and war", 'almighty dollar', 'always a bridesmaid', 'ambulance chaser', 'another day', 'ant in your pant', 'apple-pie order', 'arm and a leg', 'armchair quarterback', 'army brat', 'art imitates life', 'artsy-craftsy', 'artsy-fartsy', 'a luck would have it', 'a old a time', 'at loggerhead', 'calm before the storm', 'candle at both end', "ca n't cut the mustard", 'case of mistaken identity', 'cat out of the bag', 'cat got your tongue', 'caught red-handed', 'chapter and verse', 'checkered career', 'chicken come home to', 'roost', 'chomping at the bit', 'cleanliness is next to', 'godliness', 'clear a a bell', 'clear a mud', 'cold shoulder', 'E-ticket', 'ear to the ground', 'early bird catch the worm', 'easier said than done', 'easy a 1-2-3', 'easy a pie', 'eat crow', 'eat humble pie', 'enough already', 'even money', 'every dog hait day', 'every fiber of my being', 'everything but the kitchen sink', 'evil twin', 'existential angst', 'expert agree', 'eye for an eye', 'hair of the dog', 'hard to believe', 'have a nice day', 'head honcho', "heart 's content", 'hell-bent for leather', 'hidden agenda', 'high and the mighty', 'high on the hog', 'hold a candle to', 'hold your horse', 'hold your tongue', 'hook or by crook', 'horse of a different color', 'hot knife through butter', 'how go the battle ?', 'keep an eye on you', 'keep it down', 'keep it simple', 'keep up with the Joneses', 'keep your card close to vest', 'keep your chin up', 'keep your finger crossed', 'keep your powder dry', 'kick ass', 'kickbutt', 'kick the bucket', 'kick up your heel', 'kick you to the curb', 'kick your foot up', 'kid in a candy store', 'kill two bird with one stone', "life 's a bitch", 'lighten up', 'light out', 'like a sore thumb', 'like butter', 'like the plague', "like there 's no tomorrow", "lion 's share", 'litmus test', 'little black book', 'live and learn', 'long and short of it', 'long lost love', 'look before you leap', 'lounge lizard', 'loved and lost', 'low man on the totem pole', 'luck of the draw', 'luck of the Irish', 'off the wagon', 'old college try', 'old meet new', 'older and wiser', 'older than dirt', 'older than Methuselah', 'on the bandwagon', 'on the nose', 'on the wagon', 'on thin ice', 'one born every minute', 'one foot in the grave', 'one in a million', 'only game in town', 'only to be met', 'out of pocket', 'out of the frying pan', 'out on a limb', 'safe than sorry', 'salt of the earth', 'save face', 'scared stiff', 'scared to death', "school 's out", 'screaming meemies', 'sensreel', 'set the record straight', 'shake a stick should of', 'shoulder to the wheel', 'shouldered his way', 'shut your trap', 'sigh of relief', 'significant other', 'silence is golden', 'take one for the team', 'take the bull by the horn', 'take the plunge', 'take one to know one', 'talk turkey', 'ten foot pole', 'the earth moved', 'the final analysis', 'the real McCoy', 'the same old story', 'these thing happen', 'thick a thief', 'think outside of the box', "third time 's the charm", 'under the gun', 'under the same roof', 'understated elegance', 'unexpected twist', 'until the cow come home', 'up his sleeve', 'up the creek', 'up the wrong tree', 'very real concern', 'view with alarm', 'babe in the wood', 'back against the wall', 'back in the saddle', 'back to square one', 'back to the drawing board', 'bad to the bone', 'badge of honor', 'Badonkadonk', 'ballpark figure', 'ball to the wall', 'baptism of fire', 'bare bone', 'bark is worse than the bite', 'bark up the wrong tree', 'bat out of hell', 'bat in the belfry', 'battle royal', 'beat around the bush', 'beat the bush', 'beat me', 'behind the eight ball', 'bent out of shape', 'best foot forward', 'bet your bottom dollar', 'better half', 'better late than never', 'better mousetrap', 'better safe than sorry', 'better than ever', 'better the Devil you know', 'between a rock and a hard place', 'beyond the pale', 'bib and tucker', 'big a life', 'big fish in a small pond', 'big man on campus', 'bigger they are', 'communist conspiracy', 'conniption fit', 'could care le', "could n't care le", "could n't get to first base", 'count your blessing', 'countless hour', 'creature comfort', 'crime in the street', 'curiosity killed the cat', 'curry favor', 'cut a fine figure', 'cut and dried', 'cut to the chase', 'cut to the quick', 'cute a a button', 'fact of life', 'fair-haired

one', 'fair weather friend', 'fall off of a turnip truck', 'fat slob', 'favor u with a song', 'fear and loathing', 'feather your nest', 'fellow traveler', 'few and far between', 'field this one', 'fifteen minute of fame', 'fish nor fowl', 'fly by night', 'fly the coop', 'for the bird', 'fox in the henhouse', 'freudian slip', 'fun and game', 'fun in the sun', 'I beg to differ', 'if the shoe fit', "I 'm okay", 'in a nutshell', 'in a pinch', 'in a wink', "in harm 's way", 'in the tank', 'in your dream', 'in your face', 'inexorably drawn', 'info dump', 'influence peddling', 'intent and purpose', 'it wa a dark and stormy night', "it wo n't fly", "King 's English", "king 's ransom", 'kiss and tell', 'kiss as', 'kiss of death', 'kit and kaboodle', 'knee-high to a grasshopper', 'knock it out of the park', 'knock on wood', 'knock your sock off', 'knocked up', 'know him from Adam', 'know the rope', 'know the score', 'knuckle down', 'knuckle sandwich', 'knuckle under', 'make my day', 'male chauvinist', "man 's best friend", 'many moon', 'many-splendored thing', 'mark my word', 'meaningful relationship', 'mellow out', 'moment of glory', "moment 's respite", 'Monday morning quarterback', 'monkey suit', 'monkey see', 'motherhood and apple pie', 'mover and shaker', 'moving experience', 'my two cent', "p 's and q 's", 'pain and suffering', 'panic button', 'party pooper', 'patter of little foot', 'pas the sniff test', 'pay through the nose', 'pea in a pod', 'perfect storm', 'pig in a poke', 'pillar of society', 'pinko', 'plenty of fish in the sea', 'poison pen', 'poor a a churchmouse', 'poor excuse for', 'pot calling the kettle black', 'proud possessor', 'put my foot down', 'put your foot down', 'quick a a bunny', 'quick and the dead', 'slept like a log', 'small world', 'snake in the grass', 'snow job', 'snug a a bug', 'some of my best friend', 'something the cat dragged in', 'spade a spade', 'spare the rod', 'spitting image', 'spring to life', 'this day and age', 'this point in time', "three strike and you 're out", 'through the grapevine', 'throw in the towel', 'tiger by the tail', 'till the fat lady sings', 'time and time again', 'time is of the essense', 'tip of the iceberg', "'t is the season", 'to err is human', 'to the best of my knowledge', 'wakeup call', 'wa my face red', 'watch your tongue', 'web of intrigue', 'week of sunday', 'what a bummer', 'what come around', 'what the cat dragged in', 'what the dickens', 'what the heck', 'what the hell', 'what you see is what you get', "what 's not to like", 'wheeler-dealer', 'when in doubt', 'when push come to shove', 'when rubber meet the road', "when the cat 's away", 'when the going get tough', 'bird in the hand', 'bird and the bee', 'bird of the feather', 'bite the dust', 'bite your tongue', 'bitter disappointment', 'black a coal', 'blast from the past', 'bleeding heart', 'blind a a bat', 'blood is thicker than water', 'blood money', 'blood on your hand', 'blood sweat and tear', 'blow this pop stand', 'blow this joint', 'blushing bride', 'boil it down to', 'bone of contention', 'booze and broad', 'bored to tear', 'born and raised', 'born with a silver spoon in your mouth', 'born yesterday', 'bottom line', 'brain drain', 'brain dump', 'brass tack', 'bring home the bacon', 'broken record', "brother 's keeper", 'bull by the horn', 'bull in a china shop', 'bump in the night', 'busy a a bee', 'but seriously', 'by and large', 'darkest before the dawn', 'dead a a doornail', 'death and destruction', 'death and tax', "death 's doorstep", 'devil is in the detail', 'dim view', 'dog day', 'dog in the manger', "do n't count your chicken before they 're hatched", "do n't do the crime if you ca n't do the time", 'doubting Thomas', 'down and dirty', 'down in the dump', 'down pat', 'down the toilet', 'down the drain', 'down the hatch', 'down to earth', 'drive you up a wall', 'dutch uncle', 'dyed in the wool', 'garbage in', 'get the sack', 'get your groove back', 'get my goat', 'gift horse in the mouth', 'gilding the lily', 'give a dam n', 'give me a break', 'give me the creep', 'go him one better', 'go without sayin g', 'good deed for the day', 'good time wa had by all', 'greek to me', 'green thumb', 'green-eyed monster', 'grist for the mill', 'guiding light', 'Jack of all trade', 'jockey for position', 'Johnny-come-lately', 'joined at the hip', 'jump down your throat', 'jump in with both foot', 'jump on the bandwagon', 'jump the gun', 'jump his bone', 'jump her bone', 'junk in the trunk', 'jury is still out', 'justice is blind', 'labor of love', 'lap of luxury', 'last but not least', 'last-ditch effort', 'last hurrah', 'law of the jungle', 'law of the land', 'lay down the law', 'leap and

bound', 'let sleeping dog lie', 'let the cat out of the bag', "let 's split", 'liber
al medium', 'lie like a rug', 'life and limb', 'life imitates art', 'neat a a pin',
'needle to say', 'nip it in the bud', 'no gut', 'no love lost', 'no pain', 'no stone
unturned', 'no time like the present', 'nose to the grindstone', 'not in my back yar
d', 'not on your tintype', 'number one fan', 'numerous to mention', 'radical chic',
'rag to rich', 'raining bucket', 'raining cat and dog', 'rank and file', 'read my li
p', 'red herring', 'redheaded stepchild', 'reign supreme', 'remember the alamo', 'ro
ad to hell is paved with good intention', 'rob Peter to pay Paul', 'rock and a hard
place', 'rocket scientist', 'rocket science', 'rope a dope', 'run it up the flagpol
e', 'running dog lackey', 'squeaky wheel get the oil', 'squeaky wheel get the greas
e', 'start from scratch', 'stick in the mud', 'stick in your craw', 'still water run
deep', 'stop and smell the rose', 'store bought', 'stranger than fiction', "straw th
at broke the camel 's back", 'stubborn a a mule', 'stuff that dream are made of', 's
tuffed shirt', 'tongue-in-cheek', 'too hot to handle', 'touch of blarney', 'tough a
nail', 'tough luck', 'tough row to hoe', 'traditional family value', 'trial and trib
ulation', 'tried and true', 'trip down memory lane', 'true blue', 'turn your smile u
pside-down', 'turn your frown upside-down', 'twist of fate', 'twist and turn', 'two
to tango', 'who ha everything', 'whole ball of wax', 'whole hog', 'whole nine yard',
'whole other story', 'wild goose chase', 'wild oat', 'will wonder never cease', 'wim
p', 'win friend and influence', 'people', 'win one for the Gipper', 'wnning is every
thing', 'wisdom of the age', 'without benefit of clergy', 'wolf at the door', 'word
fail', 'work like a dog', 'worst nightmare', 'wrong side of the bed']
['abodement', 'abram', 'abroach', 'abrook', 'absey-book', 'aby', 'accdant', 'accit
e', 'accomplement', 'accompt', 'accouter', 'accoutre', 'Acheron', 'acknow', 'aconitu
m', 'acquittance', 'acture', 'Actus Primus', 'Actus Quartus', 'Actus Quintus', 'Actu
s Secundus', 'Actus Tertius', 'addle', 'adieu', 'adoptious', 'advocation', 'aery',
'afeared', 'affiance', 'affined', 'affy', 'agate', 'agazed', 'aglet', 'agood', 'agu
e', 'aiery', 'alack', 'alarum', 'Albion', 'alderliefest', 'alder-liefest', 'alder-li
evest', 'alewife', 'ale-wife', 'allayment', 'allegiant', 'allottery', 'Almain', 'alm
s drink', 'almsdrink', 'almshouse', 'amain', 'amerce', 'ames-ace', 'amort', 'anatomi
ze', 'ancient', 'a-night', 'annexment', 'anon', 'antagonist', 'antechamber', 'Anthro
pophagi', 'Anthropophaginian', 'antic', 'antick', 'anticked', 'antre', 'appeach', 'a
ppellant', 'apperil', 'apple-john', 'approof', 'apricock', 'aqua vitae', 'aqua-vita
e', 'argal', 'Argier', 'argosy', 'aright', 'armigero', 'aroint', 'a-row', 'arrant',
'arras', 'aside', 'asinego', 'assay', 'assemblance', 'assinego', 'assubjugate', 'ato
mies', 'attaint', 'attainture', 'atwain', 'a-twain', 'audit', 'auld', 'avaunt', 'avo
uch', 'aye', 'bacare', 'baccare', 'bacchanal', 'backare', 'baffle', 'baggage', 'Baja
zet', 'balcony in a Shakespearean theater', 'ballow', 'balsamum', 'ban', 'bandog',
'ban-dog', 'banns', 'Barbary', 'Barbason', 'barbermonger', 'barber-monger', 'barm',
'barnes', 'barricado', 'basilisk', 'basta', 'bastard', 'bastardy', 'bastinado', 'bat
e', 'bat-fowling', 'batler', 'batlet', 'battaille', 'battalia', 'bavin', 'bawbling',
'bawcock', 'bawdry', 'beadsman', 'beam', 'bearing cloth', 'bearing-cloth', 'beaver',
'bed of Ware', 'bed-hangings', 'bedlam', 'beetle', 'befall', 'begirt', 'behoveful',
'beldam', 'belike', 'bemadding', 'bemete', 'be-mete', 'bemoil', 'be-moil', 'be-nette
d', 'benison', 'Bergomask', 'bescreened', 'be-screened', 'beseech', 'beshrew', 'besl
ubber', 'besort', 'bespake', 'bespeak', 'bespoke', 'bestraught', 'betake', 'beteem',
'betid', 'betide', 'betimes', 'betossed', 'betwixt', 'bewray', 'bezonian', 'biding',
'biggen', 'bilberry', 'bilbo', 'bill', 'birdbolt', 'bird-bolt', 'bisson', 'blench',
'bob', 'bodement', 'bodge', 'Bodkin', 'boÓtier vert', 'bolster', 'bolted', 'bolter',
'bolting hutch', 'bolting-hutch', 'bombard', 'bona roba', 'bona-roba', 'bondman', 'b
onny', 'boot', 'bootless', 'bosky', 'botcher', 'bot', 'bottled spider', 'bounden',
'bourn', 'brabble', 'brach', 'brainish', 'brainpan', 'brain-pan', 'brake', 'braver
y', 'breese', 'Bretagne', 'brinded', 'brinish', 'Britaine', 'Brittany', 'brock', 'br
ogue', 'bruit', 'bubukle', 'buckler', 'buckram', 'bum', 'bung', 'burden a a music te
rm', 'burgonet', 'burthen', 'buskin', 'busky', 'bus', 'buzzer', 'by-room', 'cacodaem
on', 'cacodemon', 'caddis', 'caitiff', 'Calipolis', 'caliver', 'callat', 'callet',

'Cam', 'cambric', 'camlet', 'canakin', 'canary', 'canis', 'cannikin', 'canzonet', 'c
ap-a-pie', 'capon', 'carbuncle', 'carcanet', 'cardmaker', 'carlot', 'carract', 'car
t', 'carviare', 'casing', 'casque', 'cataplasm', 'catch', 'catchword', 'cater', 'cat
er-cousin', 'caterwauling', 'cautel', 'cautelous', 'caviary', 'cavil', 'cellarage',
'cellar-age', 'century', 'cerecloth', 'cerement', 'certes', 'chace', 'champain', 'ch
anticleer', 'chantry', 'chape', 'chapfallen', "chap-fall'n", 'chapless', 'chapman',
'chapt', 'charactery', 'charneco', 'charnel house', 'charnel-house', 'cheveril', 'ch
id', 'chine', 'chirurgeon', 'chopine', 'chorus', 'chough', 'chuck', 'cicatrice', 'ci
nque-pace', 'Cinque-ports', 'circummure', 'civet', 'clackdish', 'clack-dish', 'clep
e', 'clept', 'clew', 'climature', 'clinquant', 'clog', 'cloistress', 'clotpoll', 'cl
out', 'clouted brogue', 'clovest', 'cloyless', 'cockatrice', "cockerel 's stone", 'c
ockle', 'cockle hat', 'cockleshell', 'cockney', 'cock-shut time', 'codpiece', 'coffe
r', 'cog', 'cogging', 'coign', 'coil', 'collop', 'combinate', 'complice', 'complot',
'compulsative', 'con', 'concernancy', 'concupy', 'condign', 'coney', 'confixed', "co
nge 'd", 'congeed', 'conger', 'congied', 'conjunctive', 'conjuration', 'conserve',
'consort', 'conspectuity', "constring 'd", 'constringed', 'contemn', 'continuate',
'contumeliously', 'conventicle', 'convive', 'cony', "copp 'd", 'copped', 'coppice',
'coram', 'coranto', 'coronal', 'corse', 'corselet', 'costard', 'costermonger', 'cotq
uean', 'coulter', 'couplement', 'covert', 'coy', 'coystril', 'coystrill', 'coz', 'co
zen', 'cozenage', 'cozier', 'crare', 'crescive', 'crotchet', 'crown', 'crownet', 'cr
upper', 'crusado', 'cruzado', 'cubiculo', 'cuckold', 'cuckoo flower', 'cullion', 'da
ce', 'daff', 'dam', 'damosella', 'damson', 'Danskers', 'darnel', 'darraign', 'dauber
y', 'dauphin', 'debile', 'debosh', 'defeature', 'deflower', 'defunct', 'delated', 'd
elation', 'demi-wolf', 'denay', 'denier', 'denotement', 'depute', 'descant', 'design
ments', 'despatch', 'despite', 'determinate', 'diadem', 'dialogue', 'diaper', 'dic
h', 'dirge', 'disannul', 'disme', 'dispark', 'dispatch', 'disprized', 'disproperty',
'disquantity', 'dissentious', 'distaff', 'distained', 'distemperature', 'distract',
'distrain', 'ditch-dog', 'dive-dapper', 'diver', 'doit', 'dole', 'dormouse', 'dotan
t', 'doublet', 'dout', 'dower', 'dowlas', 'dowle', 'down-gyved', 'Downs', 'drab', 'd
rachma', 'draff', 'dram', 'dramatis persona', 'drave', 'drawer', 'drossy', 'drumbl
e', 'dry-beat', 'dryfoot', 'dudgeon', 'duello', 'dug', 'dump', 'dup', 'durance', 'du
rst', 'eale', 'eaning', 'eftest', 'eftsoons', 'eisel', 'eld', 'embarquement', 'embas
sage', 'emblaze', 'embossed', 'embowel', 'embracement', 'empiricutic', 'emulous', 'e
ncave', 'enchase', 'encloud', 'encompassment', 'enfeoff', 'engirt', 'englut', 'enme
w', 'enow', 'enrank', 'enroot', 'enround', 'entail', 'enter', 'enwombed', 'epilogu
e', 'epitheton', 'equipage', 'ere', 'erewhile', 'errant', 'erst', 'escot', 'esperanc
e', 'espial', 'essay', 'estridge', 'eterne', 'Ethiop', 'Ethiope', 'ever yet', 'every
et', 'exchequer', 'excursion', 'exeunt', 'exion', 'exit', 'exposture', 'exsufflicat
e', 'extirp', 'extraught', 'eyas', 'eyne', 'eyrie', 'facinorous', 'factionary', 'fad
ge', 'fain', 'fait', 'falchion', 'fallow', 'falsing', 'fap', 'farced', 'fardel', 'fa
rthing', 'fatigate', 'fay', 'fealty', 'feat', 'fecks', 'federary', "fee 'd", 'feede
r', 'fee-grief', 'Fee-simple', 'felly', 'feodary', 'fere', 'festinately', 'fico', 'f
ie', 'fillip', 'filly foal', 'fine sovereign', 'firago', 'firk', 'fitchew', 'flapdra
gon', 'fleer', 'fleshment', 'flews', 'flexure', 'flirt-gill', 'flock', 'fob', 'fob o
ff', 'foh', 'foil', 'foin', 'foison', 'fool', 'foot', 'footboy', 'foot-cloth mule',
'footing', 'fordo', 'forsooth', 'fortnight', 'fosset', 'foughten', 'foul paper', 'fo
xship', 'fracted', 'frank', 'fraught', 'fraughtage', 'frippery', 'frontlet', 'frowar
d', 'frush', 'fullam', 'fumiter', 'fumitory', 'fust', 'fustilarian', 'gaberdine', 'g
ad', 'gage', 'gainsay', 'gallery', 'galliard', 'gallia', 'galloglasses', 'gallow',
'gallowglasses', 'gamesome', 'gammon', 'gaoler', 'garboil', 'garner', 'gaskin', 'gas
ted', 'gawd', 'geck', 'gentle', 'gest', 'gib', 'giglot', 'gimmal', 'gimmer', 'gimmo
r', 'ging', 'gird', 'Gis', 'glaive', 'glanders', 'glave', 'gleek', 'glister', 'glos
e', 'glow-worm', 'gloze', 'go to the world', 'gobbet', 'goblin', 'godwit', 'good de
n', 'good-year', 'gorbellied', 'gord', 'gourd', 'gramercy', 'gratis', 'graymalkin',
'greave', 'greenly', 'gripe', 'grise', 'grize', 'groat', 'guardant', 'gudgeon', 'gue
rdon', 'gules', 'gull', 'gyve', 'habiliment', 'habitude', 'haggard', 'hale', 'half-c

rown', 'halfpenny', 'halidom', 'halter', 'handsaw', 'hap', "ha'penny", 'hapless', 'h
aply', 'hard-handed', 'hardiment', 'harebell', 'hare-bell', 'harlock', 'harrow', 'ha
sle', 'haste-post-haste', 'hatchment', 'haught', 'hautbois', 'hautboy', 'haviour',
'head', 'head-lugged', 'hebona', 'hectic', 'hedge', 'hedge-pig', 'hedgepriest', 'hef
t', 'hefted', 'heigh', 'helm', 'hempen', 'henchman', 'hent', 'hest', 'hewgh', 'Hiem
s', 'high cross', 'high-cross', 'hight', 'hilding', 'hindmost', 'hither', 'hizz', 'h
odge pudding', 'hodge-pudding', 'hoise', 'holidame', 'holla', 'hollaing', 'holp', 'h
oly-ales', 'honey-stalks', 'honorificabilitudinitatibus', 'hornbook', 'horn-mad', 'h
orologe', 'hothouse', 'hot-house', 'house of resort', 'howbeit', 'hox', 'hoy', 'hugg
ermugger', 'hull', 'hurly', 'hurricano', 'hurtle', 'husband', 'huswife', 'hyen', 'Hy
rcanian beast', 'I wi', 'ignomy', "'ild", 'illume', 'imbar', 'imbare', 'imbrue', 'im
manity', 'immoment', 'imp', 'impartment', 'impasted', 'impawn', 'imperceiverant', 'i
mpeticos', 'impleached', 'implorator', 'impone', 'importunacy', 'imposthume', 'impre
se', 'impressure', 'imputation', 'incarnadine', 'incensement', 'incertain', 'inchmea
l', 'inch-meal', 'inclip', 'incony', 'incorpsed', 'increase', 'indent', 'indifferenc
y', 'indigest', 'indign', 'indue', 'indurance', 'infamonize', 'inferreth', 'infortun
ate', 'inhearse', 'Inhoop', 'injointed', 'inkhorn', 'inkle', 'inscroll', "insculp
'd", 'insculped', 'insculpted', 'insculpture', "insinew 'd", 'insinewed', 'insistur
e', "insteep 'd", 'insteeped', 'intelligencer', 'intendment', 'intenible', 'intrins
e', 'inventorially', 'iron crow', 'irregulous', 'issue', 'iwis', 'jack-a-lent', 'jac
k', 'jade', 'jakes', 'jar', 'jaunce', 'jerkin', 'jester', 'jet', 'jocund', 'John-a-d
reams', 'joint-ring', 'jordan', 'jot', 'jutty', 'juvenal', 'kam', 'kecksy', 'keech',
'keel', 'Keisar', 'ken', 'keptst', 'kern', 'kersey', 'kibe', 'kicky-wicky', 'kine',
'kirtle', 'knack', 'knap', 'knight-errant', 'la', 'labras', 'lakin', 'Lammas', 'Lamm
as-tide', 'lampass', 'lank', 'lanthorn', 'lapwing', 'lass-lorn', 'lated', 'latten',
'laund', 'lazar', 'leagued', 'leather coat', 'leather-coat', 'leathern', 'leaf', 'le
e', 'leet', 'legerity', 'leiger', 'leman', "l'envoy", 'levy', 'libbard', 'lict', 'li
ef', 'liege', 'liegeman', 'lifeling', 'limbeck', 'Limbo', 'lime', 'lineal', 'lineame
nt', 'linstock', 'list', 'list', 'lither', 'loach', 'loaden', 'lob', 'lockram', 'lod
estar', 'lode-star', 'loggat', 'loggats', 'loof', 'lour', 'lout', 'lown', 'lowt', 'l
ozel', 'lubber', 'luce', 'lune', 'lustihood', 'lym', 'maggot-pie', 'magnifico', 'Mah
u', 'main', 'mained', 'makeless', 'malapert', 'malkin', 'malmsey', 'malt worm', 'mam
mer', 'mammet', 'mammock', 'manakin', 'mandragora', 'manikin', 'Manningtree', 'march
pane', 'marish', 'marl', 'Marry', 'mart', 'martinmas', 'martlemas', 'martlet', 'Mary
-buds', 'masque', 'matin', 'maugre', 'maw', 'mazzard', 'meacock', 'measles', 'medla
r', 'meed', 'meetest', 'meetness', 'mehercle', 'meiny', 'mell', 'mercatante', 'mere
d', 'mete-yard', 'metheglin', 'methinks', 'mew', 'mickle', 'milch', 'milkpaps', 'min
ikin', 'minim', 'minimus', 'mirable', 'miscarry', 'misconster', 'misprise', 'misprou
d', 'missingly', 'mis-tempered', 'mobled', 'Modo', 'moe', 'moiety', 'moldwarp', 'mom
e', 'momentany', 'monstruosity', 'montant', 'mooncalf', 'Morisco', 'morris', 'morris
-pike', 'morrow', 'mote', 'mow', 'mummy', 'muniment', 'mure', 'murrain', 'murther',
'musk rose', 'musk-rose', 'mutine', 'nayward', 'nayword', 'neaf', 'near-legged', 'ne
at', 'needly', 'neeld', 'neeze', 'neif', 'netherstocks', 'nether-stocks', 'noddle',
'nonce', 'nose', "not'st", 'nousle', 'novum', 'nowl', 'noyance', 'nuncle', 'nuthoo
k', 'obsequies', 'oeilliade', 'oeilliades', "o'ercrow", "o'erleaven", "o'er-leaven",
"o'er-raught", "o'er-read", "o'erskip", "o'ertrip", 'oneyer', 'ope', 'oppugnancy',
'orange-wife', 'ordinant', 'orgulous', 'orison', 'orthographer', 'ostler', 'ouph',
'ouphe', 'ousel-cock', 'outface', 'outsport', 'outwall', 'out-wall', 'overborne', 'o
vercrow', 'overglance', 'overleather', 'overleaven', 'over-raught', 'overread', 'ove
rskip', 'overtrip', 'overweening', 'over-weening', 'paction', 'paddock', 'pajock',
'pale', 'pall', 'palliament', 'palmy', 'palter', 'paly', 'pantler', 'panyn', 'panyn
æ', 'pap', 'paraquito', 'parcel', 'pardonnez-moi', "'parit", 'parle', 'parling', 'pa
rmaceti', 'Parthia', 'partisan', 'partizan', 'pash', 'passado', 'passymeasure', 'pas
sy-measure', 'pastern', 'patch', 'patine', 'pauca', 'pavan', 'pavane', 'pavin', 'pea
t', 'pedascule', 'peise', 'pell-mell', 'penny', 'pennyworth', 'penny-worth', 'peradv
enture', 'perdie', 'perdona-mi', 'perdu', 'perdurable', 'perdy', 'periapt', 'perpen

d', 'persever', 'perspective', 'pestiferous', 'petar', 'petard', 'pettish', 'pew-fel
low', 'pheeze', 'Phoebe', 'physic', 'Picht-hatch', 'pickthank', 'pick-thank', 'Pickt
-hatch', 'pight', 'pilchard', 'pill', 'pinfold', 'pinnace', "pin 's fee", 'pioned',
'pioner', 'pippin', 'pish', 'pismire', 'pittikin', 'placket', 'plain', 'plaining',
'planched', 'plat', 'plausive', 'pleached', 'pleb', 'plebeii', 'plurisy', 'poesy',
'point-device', 'poise', 'poke', 'politic', 'politicly', 'poll', 'polled', 'pomewate
r', 'poniard', 'pontifical', 'poor John', 'poperin', 'popinjay', 'porpentine', 'porr
inger', 'portance', 'posset', 'post horse', 'postern', 'post-horse', 'potation', 'po
tch', 'potting', 'pottle', 'poulter', 'pouncet box', 'pouncet-box', 'pound', 'powe
r', 'practisant', 'pratest', "prat'st", 'preceptial', 'prejudicate', 'prenominate',
'prentice', "'prentice", 'cardecu', 'coif', 'quait', "quart d ' ? cu", 'quat', 'quat
ch', 'quean', 'quern', 'questant', 'questrist', 'quiddit', 'quillet', 'quintain', 'q
uire', 'quittance', 'quiver', 'quoif', 'quoit', 'quondam', 'quotha', 'quoth-a', 'quo
tidian', 'rabato', 'rabbit-sucker', 'rack', 'rampallian', 'raught', 'ravin', 'rearmo
use', 'rear-mouse', 'reave', 'rebato', 'recheat', 'reck', 'recordation', 'recreant',
'red plague', 'rede', 'reechy', 'reel', 'refel', 'reft', 'reguerdon', 'rejourn', 're
lique', 'remember', 'remembrancer', 'remotion', 'repair', 'repugn', 'reputeless', 'r
eremouse', 'rere-mouse', 'reverb', 'Rhenish', 'rheum', 'rial', 'rib', 'riband', 'rig
gish', 'rigol', 'rivality', 'robustious', 'roe', 'roist', 'romage', 'ronyon', 'roo
d', 'rooky', 'ropery', 'roundure', 'rowel', 'royal', 'roynish', 'rub', 'ruddock', 'r
udesby', 'ruff', 'runagate', 'rushling', 'ruth', 'ryal', 'sackbut', 'sacring bell',
'sain', "Saint Colme 's Inch", 'saith', 'Sala', 'Salique Law', 'sallet', 'sanctuariz
e', 'sans', 'savour', 'saw', 'scab', 'scaffoldage', 'scald', 'scambling', 'scamel',
'scant', 'scantling', 'scape', "'scape", 'scarf up', 'scarfed', 'scathful', 'sconc
e', 'scotch', 'scrimer', 'scrip', 'scrowl', 'scroyle', 'scull', 'scullion', 'scurri
l', 'scutch', 'sea gown', 'sea-gown', 'sea-mark', 'seel', 'self', 'semblable', 'semb
lative', 'sennet', 'senseless-obstinate', 'sequent', 'sequester', 'serpigo', 'sess
a', 'Setebos', 'setter', 'several', 'shale', 'shard', "shark 'd", 'sharked', 'sheale
d', 'shelvy', 'shent', "sheriff 's post", 'shilling', 'shive', 'shoon', 'shotten',
'shough', 'shovel board', 'shovel-board', 'shrieve', 'shrift', 'shrive', 'shunless',
'sicle', 'signiory', 'signory', 'simple', 'simular', 'sirrah', 'sith', 'sithence',
'sixpence', 'skainsmates', 'skains-mates', 'skeins-mates', 'skimble-skamble', 'skink
er', 'skirr', 'sleided', 'sleight', 'sliding', 'slip', 'slop', 'slovenry', 'sluttis
h', 'smatch', 'smilets', 'smit', 'smooth', 'smoothing', 'Smulkin', 'smutch', 'snea
p', 'sneaping', 'sneck up', 'snuff', 'soft', 'soilure', 'solidare', 'sonties', 'soot
h', 'sortance', 'souse', 'sowl', 'sowle', 'spavin', 'speken', 'sperr', 'spilth', 'sp
irt', 'spital', 'splenitive', 'spongy', 'springe', 'springhalt', 'squier', 'squiny',
'stale', 'startingly', 'statist', 'statua', 'stave', 'stell', 'stile', 'stillitory',
'stilly', 'stoccado', 'stomacher', 'stonebow', 'stone-bow', 'stoup', 'strappado', 's
traw', 'strond', 'strucken', 'strumpet', 'subscription', 'succour', 'sumpter', 'supe
rflux', 'supervise', 'suppliance', 'supportance', 'supposal', 'sur-', 'surance', 'su
rplice', 'suspire', 'suum cuique', 'swag', 'swain', 'swashing', 'sweeting', 'Switze
r', 'swound', 'tabor', 'tabour', 'tainture', 'tallow', 'tang', 'tanned', 'targe', 'T
arpeian', 'tarre', 'tarry', 'teem', 'tench', 'tendance', 'tent', 'Termagant', 'testo
n', 'testoon', 'tetter', 'Tewkesbury mustard', 'thane', 'tharborough', 'Thassos', 't
heorick', 'thereunto', 'therewithal', 'thews', 'thine', 'thither', 'thorough', 'tho
u', "thou'dst", 'thraldom', 'thrall', 'thrasonical', 'three-man beetle', 'threepenc
e', 'threne', 'thrum', 'thrummed hat', 'thy', 'thy', 'thyself', 'tilt', 'tilter', 't
iltyard', 'tilt-yard', 'tirrit', 'tortive', 'touse', 'toy', 'traject', 'trencher',
'trollmydames', 'troll-my-dames', 'troth', 'trow', 'Troyan', 'truckle-bed', 'truepen
ny', 'trull', 'truncheon', 'trundle-tail', 'tuck', 'tucket', 'tun', 'tundish', 'tu
p', 'Turlygood', 'twangling', 'twelvemonth', 'twiggen', 'twilled', 'twink', 'twire',
'twit', 'twopence', "unanel 'd", 'unaneled', 'unbarbed', 'unbated', 'unbodied', 'unb
olted', 'unbreathed', 'uncape', 'unclew', 'uncoined', 'uncurrent', 'underfiends', 'u
nderskinker', 'under-wrought', 'uneath', 'unfeigned', 'ungenitured', 'ungravely', 'u
nhatched', "unhousel 'd", 'unhouseled', 'unlineal', 'unmanned', 'unmuzzle', 'unparag

```
oned', 'unpossessing', 'unprizable', 'unprizeable', 'unproperly', 'unseminared', "un
seminar 'd", 'unsifted', 'unsinewed', "unsinew 'd", 'unsisting', 'unstanched', 'unst
aunched', 'untemper', 'untempering', 'unthrift', 'upspring reel', 'usance', 'utis',
'vade', 'vail', 'vantbrace', 'varlet', 'vassalage', 'vastidity', 'vault', 'vault',
'vault', 'vault', 'vaultage', 'vaulty', 'vaunt-courier', 'vaward', 'vegetives', 'vel
ure', 'vendible', 'venew', 'veney', 'ventage', 'verger', 'verily', 'vesper', 'vestur
e', 'via', 'viand', 'victual', 'videlicet', 'viewless', 'villany', 'vinewedst', 'vin
newed', 'vinewed', 'virginalling', 'vizament', 'vizard', 'votary', 'vouchsafe', 'waf
tage', 'wafture', 'waggoner', 'wailful', 'wainropes', 'wain rope', 'wall-eyed', 'wal
l-newt', 'Walloon', 'wanion', 'wanned', 'wappened', 'wannion', "wann 'd", "wappen
'd", 'warrantise', 'water-galls', 'water-rug', 'water-work', 'weal', 'wealsman', 'we
ed', 'weet', 'welkin', 'Welsh hook', 'westward-ho !', 'wezand', 'wesand', 'weasand',
'wheaten', 'Wheeson', 'whelk', 'whence', "wh'er", "whe ' r", 'wherefore', 'wheyfac
e', 'whey-face', 'whiffler', 'while-ere', 'while', 'whimple', 'whipstock', 'whist',
'whiting', 'whitster', 'whittle', 'whoo-bub', 'whoobub', 'wight', 'willow garland',
'window-bars', 'windring', 'winter-ground', 'wi', 'wistly', 'withal', 'wittol', 'wom
by', 'wont', 'wont', 'wonted', 'woodbine', 'woodcock', 'woolward', 'wot', 'wrack',
'wrackful', 'writhled', 'wroth', 'yare', 'y-clad', 'yclept', 'yclad', 'ycleped', 'ye
llow', 'yeoman', 'yeoman', 'yeoman', 'yerk', 'yesty', 'yond', 'younker', 'zounds']
```

We will also convert all strings to lowercase to ensure that the matching is executed properly

In [ ]:
```
cliches_lemmatised= [cliche.lower() for cliche in cliches_lemmatised]
ancient_lemmatised = [phrase.lower() for phrase in ancient_lemmatised]

print(cliches_lemmatised)
print(ancient_lemmatised)
```

['ace in the hole', 'ace up your sleeve', 'acid test', 'airing dirty laundry', "all in a day 's work", 'all talk', 'all booster', 'all hat', 'all foam', 'all hammer', 'all icing', 'all lime and salt', 'all missile', 'all shot', 'all sizzle', 'all wax and no wick', 'all that and a bag of chip', 'all thumb', 'all wet', "all 's fair in love and war", 'almighty dollar', 'always a bridesmaid', 'ambulance chaser', 'another day', 'ant in your pant', 'apple-pie order', 'arm and a leg', 'armchair quarterback', 'army brat', 'art imitates life', 'artsy-craftsy', 'artsy-fartsy', 'a luck would have it', 'a old a time', 'at loggerhead', 'calm before the storm', 'candle at both end', "ca n't cut the mustard", 'case of mistaken identity', 'cat out of the bag', 'cat got your tongue', 'caught red-handed', 'chapter and verse', 'checkered career', 'chicken come home to', 'roost', 'chomping at the bit', 'cleanliness is next to', 'godliness', 'clear a a bell', 'clear a mud', 'cold shoulder', 'e-ticket', 'ear to the ground', 'early bird catch the worm', 'easier said than done', 'easy a 1-2-3', 'easy a pie', 'eat crow', 'eat humble pie', 'enough already', 'even money', 'every dog ha it day', 'every fiber of my being', 'everything but the kitchen sink', 'evil twin', 'existential angst', 'expert agree', 'eye for an eye', 'hair of the dog', 'hard to believe', 'have a nice day', 'head honcho', "heart 's content", 'hell-bent for leather', 'hidden agenda', 'high and the mighty', 'high on the hog', 'hold a candle to', 'hold your horse', 'hold your tongue', 'hook or by crook', 'horse of a different color', 'hot knife through butter', 'how go the battle ?', 'keep an eye on you', 'keep it down', 'keep it simple', 'keep up with the joneses', 'keep your card close to vest', 'keep your chin up', 'keep your finger crossed', 'keep your powder dry', 'kick a s', 'kickbutt', 'kick the bucket', 'kick up your heel', 'kick you to the curb', 'kick your foot up', 'kid in a candy store', 'kill two bird with one stone', "life 's a bitch", 'lighten up', 'light out', 'like a sore thumb', 'like butter', 'like the plague', "like there 's no tomorrow", "lion 's share", 'litmus test', 'little black book', 'live and learn', 'long and short of it', 'long lost love', 'look before you leap', 'lounge lizard', 'loved and lost', 'low man on the totem pole', 'luck of the draw', 'luck of the irish', 'off the wagon', 'old college try', 'old meet new', 'older and wiser', 'older than dirt', 'older than methuselah', 'on the bandwagon', 'on the nose', 'on the wagon', 'on thin ice', 'one born every minute', 'one foot in the grave', 'one in a million', 'only game in town', 'only to be met', 'out of pocket', 'out of the frying pan', 'out on a limb', 'safe than sorry', 'salt of the earth', 'save face', 'scared stiff', 'scared to death', "school 's out", 'screaming meemies', 'sens reel', 'set the record straight', 'shake a stick should of', 'shoulder to the wheel', 'shouldered his way', 'shut your trap', 'sigh of relief', 'significant other', 'silence is golden', 'take one for the team', 'take the bull by the horn', 'take the plunge', 'take one to know one', 'talk turkey', 'ten foot pole', 'the earth moved', 'the final analysis', 'the real mccoy', 'the same old story', 'these thing happen', 'thick a thief', 'think outside of the box', "third time 's the charm", 'under the gun', 'under the same roof', 'understated elegance', 'unexpected twist', 'until the cow come home', 'up his sleeve', 'up the creek', 'up the wrong tree', 'very real concern', 'view with alarm', 'babe in the wood', 'back against the wall', 'back in the saddle', 'back to square one', 'back to the drawing board', 'bad to the bone', 'badge of honor', 'badonkadonk', 'ballpark figure', 'ball to the wall', 'baptism of fire', 'bare bone', 'bark is worse than the bite', 'bark up the wrong tree', 'bat out of hell', 'bat in the belfry', 'battle royal', 'beat around the bush', 'beat the bush', 'beat me', 'behind the eight ball', 'bent out of shape', 'best foot forward', 'bet your bottom dollar', 'better half', 'better late than never', 'better mousetrap', 'better safe than sorry', 'better than ever', 'better the devil you know', 'between a rock and a hard place', 'beyond the pale', 'bib and tucker', 'big a life', 'big fish in a small pond', 'big man on campus', 'bigger they are', 'communist conspiracy', 'conniption fit', 'could care le', "could n't care le", "could n't get to first base", 'count your blessing', 'countless hour', 'creature comfort', 'crime in the street', 'curiosity killed the cat', 'curry favor', 'cut a fine figure', 'cut and dried', 'cut to the chase', 'cut to the quick', 'cute a a button', 'fact of life', 'fair-haired

one', 'fair weather friend', 'fall off of a turnip truck', 'fat slob', 'favor u with a song', 'fear and loathing', 'feather your nest', 'fellow traveler', 'few and far between', 'field this one', 'fifteen minute of fame', 'fish nor fowl', 'fly by night', 'fly the coop', 'for the bird', 'fox in the henhouse', 'freudian slip', 'fun and game', 'fun in the sun', 'i beg to differ', 'if the shoe fit', "i 'm okay", 'in a nutshell', 'in a pinch', 'in a wink', "in harm 's way", 'in the tank', 'in your dream', 'in your face', 'inexorably drawn', 'info dump', 'influence peddling', 'intent and purpose', 'it wa a dark and stormy night', "it wo n't fly", "king 's english", "king 's ransom", 'kiss and tell', 'kiss as', 'kiss of death', 'kit and kaboodle', 'knee-high to a grasshopper', 'knock it out of the park', 'knock on wood', 'knock your sock off', 'knocked up', 'know him from adam', 'know the rope', 'know the score', 'knuckle down', 'knuckle sandwich', 'knuckle under', 'make my day', 'male chauvinist', "man 's best friend", 'many moon', 'many-splendored thing', 'mark my word', 'meaningful relationship', 'mellow out', 'moment of glory', "moment 's respite", 'monday morning quarterback', 'monkey suit', 'monkey see', 'motherhood and apple pie', 'mover and shaker', 'moving experience', 'my two cent', "p 's and q 's", 'pain and suffering', 'panic button', 'party pooper', 'patter of little foot', 'pas the sniff test', 'pay through the nose', 'pea in a pod', 'perfect storm', 'pig in a poke', 'pillar of society', 'pinko', 'plenty of fish in the sea', 'poison pen', 'poor a a churchmouse', 'poor excuse for', 'pot calling the kettle black', 'proud possessor', 'put my foot down', 'put your foot down', 'quick a a bunny', 'quick and the dead', 'slept like a log', 'small world', 'snake in the grass', 'snow job', 'snug a a bug', 'some of my best friend', 'something the cat dragged in', 'spade a spade', 'spare the rod', 'spitting image', 'spring to life', 'this day and age', 'this point in time', "three strike and you 're out", 'through the grapevine', 'throw in the towel', 'tiger by the tail', 'till the fat lady sings', 'time and time again', 'time is of the essense', 'tip of the iceberg', "'t is the season", 'to err is human', 'to the best of my knowledge', 'wakeup call', 'wa my face red', 'watch your tongue', 'web of intrigue', 'week of sunday', 'what a bummer', 'what come around', 'what the cat dragged in', 'what the dickens', 'what the heck', 'what the hell', 'what you see is what you get', "what 's not to like", 'wheeler-dealer', 'when in doubt', 'when push come to shove', 'when rubber meet the road', "when the cat 's away", 'when the going get tough', 'bird in the hand', 'bird and the bee', 'bird of the feather', 'bite the dust', 'bite your tongue', 'bitter disappointment', 'black a coal', 'blast from the past', 'bleeding heart', 'blind a a bat', 'blood is thicker than water', 'blood money', 'blood on your hand', 'blood sweat and tear', 'blow this pop stand', 'blow this joint', 'blushing bride', 'boil it down to', 'bone of contention', 'booze and broad', 'bored to tear', 'born and raised', 'born with a silver spoon in your mouth', 'born yesterday', 'bottom line', 'brain drain', 'brain dump', 'brass tack', 'bring home the bacon', 'broken record', "brother 's keeper", 'bull by the horn', 'bull in a china shop', 'bump in the night', 'busy a a bee', 'but seriously', 'by and large', 'darkest before the dawn', 'dead a a doornail', 'death and destruction', 'death and tax', "death 's doorstep", 'devil is in the detail', 'dim view', 'dog day', 'dog in the manger', "do n't count your chicken before they 're hatched", "do n't do the crime if you ca n't do the time", 'doubting thomas', 'down and dirty', 'down in the dump', 'down pat', 'down the toilet', 'down the drain', 'down the hatch', 'down to earth', 'drive you up a wall', 'dutch uncle', 'dyed in the wool', 'garbage in', 'get the sack', 'get your groove back', 'get my goat', 'gift horse in the mouth', 'gilding the lily', 'give a damn', 'give me a break', 'give me the creep', 'go him one better', 'go without saying', 'good deed for the day', 'good time wa had by all', 'greek to me', 'green thumb', 'green-eyed monster', 'grist for the mill', 'guiding light', 'jack of all trade', 'jockey for position', 'johnny-come-lately', 'joined at the hip', 'jump down your throat', 'jump in with both foot', 'jump on the bandwagon', 'jump the gun', 'jump his bone', 'jump her bone', 'junk in the trunk', 'jury is still out', 'justice is blind', 'labor of love', 'lap of luxury', 'last but not least', 'last-ditch effort', 'last hurrah', 'law of the jungle', 'law of the land', 'lay down the law', 'leap and

bound', 'let sleeping dog lie', 'let the cat out of the bag', "let 's split", 'liber
al medium', 'lie like a rug', 'life and limb', 'life imitates art', 'neat a a pin',
'needle to say', 'nip it in the bud', 'no gut', 'no love lost', 'no pain', 'no stone
unturned', 'no time like the present', 'nose to the grindstone', 'not in my back yar
d', 'not on your tintype', 'number one fan', 'numerous to mention', 'radical chic',
'rag to rich', 'raining bucket', 'raining cat and dog', 'rank and file', 'read my li
p', 'red herring', 'redheaded stepchild', 'reign supreme', 'remember the alamo', 'ro
ad to hell is paved with good intention', 'rob peter to pay paul', 'rock and a hard
place', 'rocket scientist', 'rocket science', 'rope a dope', 'run it up the flagpol
e', 'running dog lackey', 'squeaky wheel get the oil', 'squeaky wheel get the greas
e', 'start from scratch', 'stick in the mud', 'stick in your craw', 'still water run
deep', 'stop and smell the rose', 'store bought', 'stranger than fiction', "straw th
at broke the camel 's back", 'stubborn a a mule', 'stuff that dream are made of', 's
tuffed shirt', 'tongue-in-cheek', 'too hot to handle', 'touch of blarney', 'tough a
nail', 'tough luck', 'tough row to hoe', 'traditional family value', 'trial and trib
ulation', 'tried and true', 'trip down memory lane', 'true blue', 'turn your smile u
pside-down', 'turn your frown upside-down', 'twist of fate', 'twist and turn', 'two
to tango', 'who ha everything', 'whole ball of wax', 'whole hog', 'whole nine yard',
'whole other story', 'wild goose chase', 'wild oat', 'will wonder never cease', 'wim
p', 'win friend and influence', 'people', 'win one for the gipper', 'wnning is every
thing', 'wisdom of the age', 'without benefit of clergy', 'wolf at the door', 'word
fail', 'work like a dog', 'worst nightmare', 'wrong side of the bed']
['abodement', 'abram', 'abroach', 'abrook', 'absey-book', 'aby', 'accdant', 'accit
e', 'accomplement', 'accompt', 'accouter', 'accoutre', 'acheron', 'acknow', 'aconitu
m', 'acquittance', 'acture', 'actus primus', 'actus quartus', 'actus quintus', 'actu
s secundus', 'actus tertius', 'addle', 'adieu', 'adoptious', 'advocation', 'aery',
'afeared', 'affiance', 'affined', 'affy', 'agate', 'agazed', 'aglet', 'agood', 'agu
e', 'aiery', 'alack', 'alarum', 'albion', 'alderliefest', 'alder-liefest', 'alder-li
evest', 'alewife', 'ale-wife', 'allayment', 'allegiant', 'allottery', 'almain', 'alm
s drink', 'almsdrink', 'almshouse', 'amain', 'amerce', 'ames-ace', 'amort', 'anatomi
ze', 'ancient', 'a-night', 'annexment', 'anon', 'antagonist', 'antechamber', 'anthro
pophagi', 'anthropophaginian', 'antic', 'antick', 'anticked', 'antre', 'appeach', 'a
ppellant', 'apperil', 'apple-john', 'approof', 'apricock', 'aqua vitae', 'aqua-vita
e', 'argal', 'argier', 'argosy', 'aright', 'armigero', 'aroint', 'a-row', 'arrant',
'arras', 'aside', 'asinego', 'assay', 'assemblance', 'assinego', 'assubjugate', 'ato
mies', 'attaint', 'attainture', 'atwain', 'a-twain', 'audit', 'auld', 'avaunt', 'avo
uch', 'aye', 'bacare', 'baccare', 'bacchanal', 'backare', 'baffle', 'baggage', 'baja
zet', 'balcony in a shakespearean theater', 'ballow', 'balsamum', 'ban', 'bandog',
'ban-dog', 'banns', 'barbary', 'barbason', 'barbermonger', 'barber-monger', 'barm',
'barnes', 'barricado', 'basilisk', 'basta', 'bastard', 'bastardy', 'bastinado', 'bat
e', 'bat-fowling', 'batler', 'batlet', 'battaille', 'battalia', 'bavin', 'bawbling',
'bawcock', 'bawdry', 'beadsman', 'beam', 'bearing cloth', 'bearing-cloth', 'beaver',
'bed of ware', 'bed-hangings', 'bedlam', 'beetle', 'befall', 'begirt', 'behoveful',
'beldam', 'belike', 'bemadding', 'bemete', 'be-mete', 'bemoil', 'be-moil', 'be-nette
d', 'benison', 'bergomask', 'bescreened', 'be-screened', 'beseech', 'beshrew', 'besl
ubber', 'besort', 'bespake', 'bespeak', 'bespoke', 'bestraught', 'betake', 'beteem',
'betid', 'betide', 'betimes', 'betossed', 'betwixt', 'bewray', 'bezonian', 'biding',
'biggen', 'bilberry', 'bilbo', 'bill', 'birdbolt', 'bird-bolt', 'bisson', 'blench',
'bob', 'bodement', 'bodge', 'bodkin', 'boótier vert', 'bolster', 'bolted', 'bolter',
'bolting hutch', 'bolting-hutch', 'bombard', 'bona roba', 'bona-roba', 'bondman', 'b
onny', 'boot', 'bootless', 'bosky', 'botcher', 'bot', 'bottled spider', 'bounden',
'bourn', 'brabble', 'brach', 'brainish', 'brainpan', 'brain-pan', 'brake', 'braver
y', 'breese', 'bretagne', 'brinded', 'brinish', 'britaine', 'brittany', 'brock', 'br
ogue', 'bruit', 'bubukle', 'buckler', 'buckram', 'bum', 'bung', 'burden a a music te
rm', 'burgonet', 'burthen', 'buskin', 'busky', 'bus', 'buzzer', 'by-room', 'cacodaem
on', 'cacodemon', 'caddis', 'caitiff', 'calipolis', 'caliver', 'callat', 'callet',

'cam', 'cambric', 'camlet', 'canakin', 'canary', 'canis', 'cannikin', 'canzonet', 'c
ap-a-pie', 'capon', 'carbuncle', 'carcanet', 'cardmaker', 'carlot', 'carract', 'car
t', 'carviare', 'casing', 'casque', 'cataplasm', 'catch', 'catchword', 'cater', 'cat
er-cousin', 'caterwauling', 'cautel', 'cautelous', 'caviary', 'cavil', 'cellarage',
'cellar-age', 'century', 'cerecloth', 'cerement', 'certes', 'chace', 'champain', 'ch
anticleer', 'chantry', 'chape', 'chapfallen', "chap-fall'n", 'chapless', 'chapman',
'chapt', 'charactery', 'charneco', 'charnel house', 'charnel-house', 'cheveril', 'ch
id', 'chine', 'chirurgeon', 'chopine', 'chorus', 'chough', 'chuck', 'cicatrice', 'ci
nque-pace', 'cinque-ports', 'circummure', 'civet', 'clackdish', 'clack-dish', 'clep
e', 'clept', 'clew', 'climature', 'clinquant', 'clog', 'cloistress', 'clotpoll', 'cl
out', 'clouted brogue', 'clovest', 'cloyless', 'cockatrice', "cockerel 's stone", 'c
ockle', 'cockle hat', 'cockleshell', 'cockney', 'cock-shut time', 'codpiece', 'coffe
r', 'cog', 'cogging', 'coign', 'coil', 'collop', 'combinate', 'complice', 'complot',
'compulsative', 'con', 'concernancy', 'concupy', 'condign', 'coney', 'confixed', "co
nge 'd", 'congeed', 'conger', 'congied', 'conjunctive', 'conjuration', 'conserve',
'consort', 'conspectuity', "constring 'd", 'constringed', 'contemn', 'continuate',
'contumeliously', 'conventicle', 'convive', 'cony', "copp 'd", 'copped', 'coppice',
'coram', 'coranto', 'coronal', 'corse', 'corselet', 'costard', 'costermonger', 'cotq
uean', 'coulter', 'couplement', 'covert', 'coy', 'coystril', 'coystrill', 'coz', 'co
zen', 'cozenage', 'cozier', 'crare', 'crescive', 'crotchet', 'crown', 'crownet', 'cr
upper', 'crusado', 'cruzado', 'cubiculo', 'cuckold', 'cuckoo flower', 'cullion', 'da
ce', 'daff', 'dam', 'damosella', 'damson', 'danskers', 'darnel', 'darraign', 'dauber
y', 'dauphin', 'debile', 'debosh', 'defeature', 'deflower', 'defunct', 'delated', 'd
elation', 'demi-wolf', 'denay', 'denier', 'denotement', 'depute', 'descant', 'design
ments', 'despatch', 'despite', 'determinate', 'diadem', 'dialogue', 'diaper', 'dic
h', 'dirge', 'disannul', 'disme', 'dispark', 'dispatch', 'disprized', 'disproperty',
'disquantity', 'dissentious', 'distaff', 'distained', 'distemperature', 'distract',
'distrain', 'ditch-dog', 'dive-dapper', 'diver', 'doit', 'dole', 'dormouse', 'dotan
t', 'doublet', 'dout', 'dower', 'dowlas', 'dowle', 'down-gyved', 'downs', 'drab', 'd
rachma', 'draff', 'dram', 'dramatis persona', 'drave', 'drawer', 'drossy', 'drumbl
e', 'dry-beat', 'dryfoot', 'dudgeon', 'duello', 'dug', 'dump', 'dup', 'durance', 'du
rst', 'eale', 'eaning', 'eftest', 'eftsoons', 'eisel', 'eld', 'embarquement', 'embas
sage', 'emblaze', 'embossed', 'embowel', 'embracement', 'empiricutic', 'emulous', 'e
ncave', 'enchase', 'encloud', 'encompassment', 'enfeoff', 'engirt', 'englut', 'enme
w', 'enow', 'enrank', 'enroot', 'enround', 'entail', 'enter', 'enwombed', 'epilogu
e', 'epitheton', 'equipage', 'ere', 'erewhile', 'errant', 'erst', 'escot', 'esperanc
e', 'espial', 'essay', 'estridge', 'eterne', 'ethiop', 'ethiope', 'ever yet', 'every
et', 'exchequer', 'excursion', 'exeunt', 'exion', 'exit', 'exposture', 'exsufflicat
e', 'extirp', 'extraught', 'eyas', 'eyne', 'eyrie', 'facinorous', 'factionary', 'fad
ge', 'fain', 'fait', 'falchion', 'fallow', 'falsing', 'fap', 'farced', 'fardel', 'fa
rthing', 'fatigate', 'fay', 'fealty', 'feat', 'fecks', 'federary', "fee 'd", 'feede
r', 'fee-grief', 'fee-simple', 'felly', 'feodary', 'fere', 'festinately', 'fico', 'f
ie', 'fillip', 'filly foal', 'fine sovereign', 'firago', 'firk', 'fitchew', 'flapdra
gon', 'fleer', 'fleshment', 'flews', 'flexure', 'flirt-gill', 'flock', 'fob', 'fob o
ff', 'foh', 'foil', 'foin', 'foison', 'fool', 'foot', 'footboy', 'foot-cloth mule',
'footing', 'fordo', 'forsooth', 'fortnight', 'fosset', 'foughten', 'foul paper', 'fo
xship', 'fracted', 'frank', 'fraught', 'fraughtage', 'frippery', 'frontlet', 'frowar
d', 'frush', 'fullam', 'fumiter', 'fumitory', 'fust', 'fustilarian', 'gaberdine', 'g
ad', 'gage', 'gainsay', 'gallery', 'galliard', 'gallia', 'galloglasses', 'gallow',
'gallowglasses', 'gamesome', 'gammon', 'gaoler', 'garboil', 'garner', 'gaskin', 'gas
ted', 'gawd', 'geck', 'gentle', 'gest', 'gib', 'giglot', 'gimmal', 'gimmer', 'gimmo
r', 'ging', 'gird', 'gis', 'glaive', 'glanders', 'glave', 'gleek', 'glister', 'glos
e', 'glow-worm', 'gloze', 'go to the world', 'gobbet', 'goblin', 'godwit', 'good de
n', 'good-year', 'gorbellied', 'gord', 'gourd', 'gramercy', 'gratis', 'graymalkin',
'greave', 'greenly', 'gripe', 'grise', 'grize', 'groat', 'guardant', 'gudgeon', 'gue
rdon', 'gules', 'gull', 'gyve', 'habiliment', 'habitude', 'haggard', 'hale', 'half-c

rown', 'halfpenny', 'halidom', 'halter', 'handsaw', 'hap', "ha'penny", 'hapless', 'h
aply', 'hard-handed', 'hardiment', 'harebell', 'hare-bell', 'harlock', 'harrow', 'ha
sle', 'haste-post-haste', 'hatchment', 'haught', 'hautbois', 'hautboy', 'haviour',
'head', 'head-lugged', 'hebona', 'hectic', 'hedge', 'hedge-pig', 'hedgepriest', 'hef
t', 'hefted', 'heigh', 'helm', 'hempen', 'henchman', 'hent', 'hest', 'hewgh', 'hiem
s', 'high cross', 'high-cross', 'hight', 'hilding', 'hindmost', 'hither', 'hizz', 'h
odge pudding', 'hodge-pudding', 'hoise', 'holidame', 'holla', 'hollaing', 'holp', 'h
oly-ales', 'honey-stalks', 'honorificabilitudinitatibus', 'hornbook', 'horn-mad', 'h
orologe', 'hothouse', 'hot-house', 'house of resort', 'howbeit', 'hox', 'hoy', 'hugg
ermugger', 'hull', 'hurly', 'hurricano', 'hurtle', 'husband', 'huswife', 'hyen', 'hy
rcanian beast', 'i wi', 'ignomy', "'ild", 'illume', 'imbar', 'imbare', 'imbrue', 'im
manity', 'immoment', 'imp', 'impartment', 'impasted', 'impawn', 'imperceiverant', 'i
mpeticos', 'impleached', 'implorator', 'impone', 'importunacy', 'imposthume', 'impre
se', 'impressure', 'imputation', 'incarnadine', 'incensement', 'incertain', 'inchmea
l', 'inch-meal', 'inclip', 'incony', 'incorpsed', 'increase', 'indent', 'indifferenc
y', 'indigest', 'indign', 'indue', 'indurance', 'infamonize', 'inferreth', 'infortun
ate', 'inhearse', 'inhoop', 'injointed', 'inkhorn', 'inkle', 'inscroll', "insculp
'd", 'insculped', 'insculpted', 'insculpture', "insinew 'd", 'insinewed', 'insistur
e', "insteep 'd", 'insteeped', 'intelligencer', 'intendment', 'intenible', 'intrins
e', 'inventorially', 'iron crow', 'irregulous', 'issue', 'iwis', 'jack-a-lent', 'jac
k', 'jade', 'jakes', 'jar', 'jaunce', 'jerkin', 'jester', 'jet', 'jocund', 'john-a-d
reams', 'joint-ring', 'jordan', 'jot', 'jutty', 'juvenal', 'kam', 'kecksy', 'keech',
'keel', 'keisar', 'ken', 'keptst', 'kern', 'kersey', 'kibe', 'kicky-wicky', 'kine',
'kirtle', 'knack', 'knap', 'knight-errant', 'la', 'labras', 'lakin', 'lammas', 'lamm
as-tide', 'lampass', 'lank', 'lanthorn', 'lapwing', 'lass-lorn', 'lated', 'latten',
'laund', 'lazar', 'leagued', 'leather coat', 'leather-coat', 'leathern', 'leaf', 'le
e', 'leet', 'legerity', 'leiger', 'leman', "l'envoy", 'levy', 'libbard', 'lict', 'li
ef', 'liege', 'liegeman', 'lifeling', 'limbeck', 'limbo', 'lime', 'lineal', 'lineame
nt', 'linstock', 'list', 'list', 'lither', 'loach', 'loaden', 'lob', 'lockram', 'lod
estar', 'lode-star', 'loggat', 'loggats', 'loof', 'lour', 'lout', 'lown', 'lowt', 'l
ozel', 'lubber', 'luce', 'lune', 'lustihood', 'lym', 'maggot-pie', 'magnifico', 'mah
u', 'main', 'mained', 'makeless', 'malapert', 'malkin', 'malmsey', 'malt worm', 'mam
mer', 'mammet', 'mammock', 'manakin', 'mandragora', 'manikin', 'manningtree', 'march
pane', 'marish', 'marl', 'marry', 'mart', 'martinmas', 'martlemas', 'martlet', 'mary
-buds', 'masque', 'matin', 'maugre', 'maw', 'mazzard', 'meacock', 'measles', 'medla
r', 'meed', 'meetest', 'meetness', 'mehercle', 'meiny', 'mell', 'mercatante', 'mere
d', 'mete-yard', 'metheglin', 'methinks', 'mew', 'mickle', 'milch', 'milkpaps', 'min
ikin', 'minim', 'minimus', 'mirable', 'miscarry', 'misconster', 'misprise', 'misprou
d', 'missingly', 'mis-tempered', 'mobled', 'modo', 'moe', 'moiety', 'moldwarp', 'mom
e', 'momentany', 'monstruosity', 'montant', 'mooncalf', 'morisco', 'morris', 'morris
-pike', 'morrow', 'mote', 'mow', 'mummy', 'muniment', 'mure', 'murrain', 'murther',
'musk rose', 'musk-rose', 'mutine', 'nayward', 'nayword', 'neaf', 'near-legged', 'ne
at', 'needly', 'neeld', 'neeze', 'neif', 'netherstocks', 'nether-stocks', 'noddle',
'nonce', 'nose', "not'st", 'nousle', 'novum', 'nowl', 'noyance', 'nuncle', 'nuthoo
k', 'obsequies', 'oeilliade', 'oeilliades', "o'ercrow", "o'erleaven", "o'er-leaven",
"o'er-raught", "o'er-read", "o'erskip", "o'ertrip", 'oneyer', 'ope', 'oppugnancy',
'orange-wife', 'ordinant', 'orgulous', 'orison', 'orthographer', 'ostler', 'ouph',
'ouphe', 'ousel-cock', 'outface', 'outsport', 'outwall', 'out-wall', 'overborne', 'o
vercrow', 'overglance', 'overleather', 'overleaven', 'over-raught', 'overread', 'ove
rskip', 'overtrip', 'overweening', 'over-weening', 'paction', 'paddock', 'pajock',
'pale', 'pall', 'palliament', 'palmy', 'palter', 'paly', 'pantler', 'panyn', 'panyn
æ', 'pap', 'paraquito', 'parcel', 'pardonnez-moi', "'parit", 'parle', 'parling', 'pa
rmaceti', 'parthia', 'partisan', 'partizan', 'pash', 'passado', 'passymeasure', 'pas
sy-measure', 'pastern', 'patch', 'patine', 'pauca', 'pavan', 'pavane', 'pavin', 'pea
t', 'pedascule', 'peise', 'pell-mell', 'penny', 'pennyworth', 'penny-worth', 'peradv
enture', 'perdie', 'perdona-mi', 'perdu', 'perdurable', 'perdy', 'periapt', 'perpen

d', 'persever', 'perspective', 'pestiferous', 'petar', 'petard', 'pettish', 'pew-fel
low', 'pheeze', 'phoebe', 'physic', 'picht-hatch', 'pickthank', 'pick-thank', 'pickt
-hatch', 'pight', 'pilchard', 'pill', 'pinfold', 'pinnace', "pin 's fee", 'pioned',
'pioner', 'pippin', 'pish', 'pismire', 'pittikin', 'placket', 'plain', 'plaining',
'planched', 'plat', 'plausive', 'pleached', 'pleb', 'plebeii', 'plurisy', 'poesy',
'point-device', 'poise', 'poke', 'politic', 'politicly', 'poll', 'polled', 'pomewate
r', 'poniard', 'pontifical', 'poor john', 'poperin', 'popinjay', 'porpentine', 'porr
inger', 'portance', 'posset', 'post horse', 'postern', 'post-horse', 'potation', 'po
tch', 'potting', 'pottle', 'poulter', 'pouncet box', 'pouncet-box', 'pound', 'powe
r', 'practisant', 'pratest', "prat'st", 'preceptial', 'prejudicate', 'prenominate',
'prentice', "'prentice", 'cardecu', 'coif', 'quait', "quart d ' ? cu", 'quat', 'quat
ch', 'quean', 'quern', 'questant', 'questrist', 'quiddit', 'quillet', 'quintain', 'q
uire', 'quittance', 'quiver', 'quoif', 'quoit', 'quondam', 'quotha', 'quoth-a', 'quo
tidian', 'rabato', 'rabbit-sucker', 'rack', 'rampallian', 'raught', 'ravin', 'rearmo
use', 'rear-mouse', 'reave', 'rebato', 'recheat', 'reck', 'recordation', 'recreant',
'red plague', 'rede', 'reechy', 'reel', 'refel', 'reft', 'reguerdon', 'rejourn', 're
lique', 'remember', 'remembrancer', 'remotion', 'repair', 'repugn', 'reputeless', 'r
eremouse', 'rere-mouse', 'reverb', 'rhenish', 'rheum', 'rial', 'rib', 'riband', 'rig
gish', 'rigol', 'rivality', 'robustious', 'roe', 'roist', 'romage', 'ronyon', 'roo
d', 'rooky', 'ropery', 'roundure', 'rowel', 'royal', 'roynish', 'rub', 'ruddock', 'r
udesby', 'ruff', 'runagate', 'rushling', 'ruth', 'ryal', 'sackbut', 'sacring bell',
'sain', "saint colme 's inch", 'saith', 'sala', 'salique law', 'sallet', 'sanctuariz
e', 'sans', 'savour', 'saw', 'scab', 'scaffoldage', 'scald', 'scambling', 'scamel',
'scant', 'scantling', 'scape', "'scape", 'scarf up', 'scarfed', 'scathful', 'sconc
e', 'scotch', 'scrimer', 'scrip', 'scrowl', 'scroyle', 'scull', 'scullion', 'scurri
l', 'scutch', 'sea gown', 'sea-gown', 'sea-mark', 'seel', 'self', 'semblable', 'semb
lative', 'sennet', 'senseless-obstinate', 'sequent', 'sequester', 'serpigo', 'sess
a', 'setebos', 'setter', 'several', 'shale', 'shard', "shark 'd", 'sharked', 'sheale
d', 'shelvy', 'shent', "sheriff 's post", 'shilling', 'shive', 'shoon', 'shotten',
'shough', 'shovel board', 'shovel-board', 'shrieve', 'shrift', 'shrive', 'shunless',
'sicle', 'signiory', 'signory', 'simple', 'simular', 'sirrah', 'sith', 'sithence',
'sixpence', 'skainsmates', 'skains-mates', 'skeins-mates', 'skimble-skamble', 'skink
er', 'skirr', 'sleided', 'sleight', 'sliding', 'slip', 'slop', 'slovenry', 'sluttis
h', 'smatch', 'smilets', 'smit', 'smooth', 'smoothing', 'smulkin', 'smutch', 'snea
p', 'sneaping', 'sneck up', 'snuff', 'soft', 'soilure', 'solidare', 'sonties', 'soot
h', 'sortance', 'souse', 'sowl', 'sowle', 'spavin', 'speken', 'sperr', 'spilth', 'sp
irt', 'spital', 'splenitive', 'spongy', 'springe', 'springhalt', 'squier', 'squiny',
'stale', 'startingly', 'statist', 'statua', 'stave', 'stell', 'stile', 'stillitory',
'stilly', 'stoccado', 'stomacher', 'stonebow', 'stone-bow', 'stoup', 'strappado', 's
traw', 'strond', 'strucken', 'strumpet', 'subscription', 'succour', 'sumpter', 'supe
rflux', 'supervise', 'suppliance', 'supportance', 'supposal', 'sur-', 'surance', 'su
rplice', 'suspire', 'suum cuique', 'swag', 'swain', 'swashing', 'sweeting', 'switze
r', 'swound', 'tabor', 'tabour', 'tainture', 'tallow', 'tang', 'tanned', 'targe', 't
arpeian', 'tarre', 'tarry', 'teem', 'tench', 'tendance', 'tent', 'termagant', 'testo
n', 'testoon', 'tetter', 'tewkesbury mustard', 'thane', 'tharborough', 'thassos', 't
heorick', 'thereunto', 'therewithal', 'thews', 'thine', 'thither', 'thorough', 'tho
u', "thou'dst", 'thraldom', 'thrall', 'thrasonical', 'three-man beetle', 'threepenc
e', 'threne', 'thrum', 'thrummed hat', 'thy', 'thy', 'thyself', 'tilt', 'tilter', 't
iltyard', 'tilt-yard', 'tirrit', 'tortive', 'touse', 'toy', 'traject', 'trencher',
'trollmydames', 'troll-my-dames', 'troth', 'trow', 'troyan', 'truckle-bed', 'truepen
ny', 'trull', 'truncheon', 'trundle-tail', 'tuck', 'tucket', 'tun', 'tundish', 'tu
p', 'turlygood', 'twangling', 'twelvemonth', 'twiggen', 'twilled', 'twink', 'twire',
'twit', 'twopence', "unanel 'd", 'unaneled', 'unbarbed', 'unbated', 'unbodied', 'unb
olted', 'unbreathed', 'uncape', 'unclew', 'uncoined', 'uncurrent', 'underfiends', 'u
nderskinker', 'under-wrought', 'uneath', 'unfeigned', 'ungenitured', 'ungravely', 'u
nhatched', "unhousel 'd", 'unhouseled', 'unlineal', 'unmanned', 'unmuzzle', 'unparag

```
oned', 'unpossessing', 'unprizable', 'unprizeable', 'unproperly', 'unseminared', "un
seminar 'd", 'unsifted', 'unsinewed', "unsinew 'd", 'unsisting', 'unstanched', 'unst
aunched', 'untemper', 'untempering', 'unthrift', 'upspring reel', 'usance', 'utis',
'vade', 'vail', 'vantbrace', 'varlet', 'vassalage', 'vastidity', 'vault', 'vault',
'vault', 'vault', 'vaultage', 'vaulty', 'vaunt-courier', 'vaward', 'vegetives', 'vel
ure', 'vendible', 'venew', 'veney', 'ventage', 'verger', 'verily', 'vesper', 'vestur
e', 'via', 'viand', 'victual', 'videlicet', 'viewless', 'villany', 'vinewedst', 'vin
newed', 'vinewed', 'virginalling', 'vizament', 'vizard', 'votary', 'vouchsafe', 'waf
tage', 'wafture', 'waggoner', 'wailful', 'wainropes', 'wain rope', 'wall-eyed', 'wal
l-newt', 'walloon', 'wanion', 'wanned', 'wappened', 'wannion', "wann 'd", "wappen
'd", 'warrantise', 'water-galls', 'water-rug', 'water-work', 'weal', 'wealsman', 'we
ed', 'weet', 'welkin', 'welsh hook', 'westward-ho !', 'wezand', 'wesand', 'weasand',
'wheaten', 'wheeson', 'whelk', 'whence', "wh'er", "whe ' r", 'wherefore', 'wheyfac
e', 'whey-face', 'whiffler', 'while-ere', 'while', 'whimple', 'whipstock', 'whist',
'whiting', 'whitster', 'whittle', 'whoo-bub', 'whoobub', 'wight', 'willow garland',
'window-bars', 'windring', 'winter-ground', 'wi', 'wistly', 'withal', 'wittol', 'wom
by', 'wont', 'wont', 'wonted', 'woodbine', 'woodcock', 'woolward', 'wot', 'wrack',
'wrackful', 'writhled', 'wroth', 'yare', 'y-clad', 'yclept', 'yclad', 'ycleped', 'ye
llow', 'yeoman', 'yeoman', 'yeoman', 'yerk', 'yesty', 'yond', 'younker', 'zounds']
```

In [ ]:
```
#initialise empty columns for the features
df_cleaned['cliche_count'] = 0
df_cleaned['ancient_phrase_count'] = 0
```

In [ ]:
```
for i in range(0, len(df_cleaned)):
    answer = df_cleaned.at[i, 'answer'] #retrieve answer from df
    answer = answer.lower() # convert the answer to all lower case

    cliche_cntr = 0
    for cliche in cliches:
        if cliche in answer:
            cliche_cntr += 1

    ancient_phrase_cntr = 0
    for phrase in ancient_phrases:
        if phrase in answer:
            ancient_phrase_cntr += 1

    df_cleaned.at[i, 'cliche_count'] = cliche_cntr
    df_cleaned.at[i, 'ancient_phrase_count'] = ancient_phrase_cntr
```

In [ ]:
```
df_cleaned.sample(3)
```

Out[ ]:

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	question	answer	is_human	filtered_answer	le
16676	16676	16676	16676	How is Butterfly Trade Strategy good if the mi...	The butterfly spread is a neutral options trad...	0	The butterfly spread neutral option trading st...	T
29588	29588	29588	29588	How to start a business ? I 'm starting a cons...	Starting a business can be a complex process, ...	0	Starting business complex process , also rewar...	
41236	41236	41236	41236	Why do we like to sleep with a hand under our ...	Sleeping with a hand under your head is a habi...	0	Sleeping hand head habit people develop feel c...	

3 rows × 26 columns

We will also save the final dataframe with all extracted features to a csv file

In [ ]:
```
df_cleaned.to_csv('final_feature_data.csv')
```

3. Classification

3.1 Model Testing

As mentioned in the report, we will be testing and evaluating 3 different classification algorithms, implemented using the Scikit-learn python package - Support Vector Machines (SVM), Logistic Regression (LR), and Random Forest Classifier (RFC).

Of greatest interest is the SVM algorithm, as testing by Fröhling and Zubiaga [2021] and Nguyen-Son et al. [2017] has shown that Support Vector Machines (SVM) seem to perform the best for classifying LLM-generated texts. Specifically, Nguyen-Son et al. [2017] found that using an SVM with the Stochastic Gradient Decent (SGD) algorithm achieved the best results.

As such, we will be testing a using an SVM with the SGD algorithm, termed as SVM(SGD). This will be implemented using the SGD classifer (SGDC) with the 'hinge' loss function (hinge uses a linear SVM) available in the scikit-learn package.

First, install scikit-learn and import the neccessary packages and load the csv file of extracted features

In [ ]:
```
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
df = pd.read_csv('final_feature_data.csv')
```

We will also drop some of the unused index columns that were created previously when saving the intermediate feature data to csv file

In [ ]:
```
df.drop(df.columns[0:4], axis=1, inplace=True)
```

In [ ]:
```
df.sample(3)
```

Out[ ]:

	question	answer	is_human	filtered_answer	lemmatised_answer	word_count	s
15911	What makes us stop from rolling out of bed whi...	When we sleep, our bodies are in a relaxed sta...	0	When sleep , body relaxed state muscle tense a...	When we sleep , our body are in a relaxed stat...	189	
20062	If blood travels through veins , why is there ...	It 's also in capillaries . Capillaries are ri...	1	It 's also capillary . Capillaries riddled bod...	It 's also in capillary . Capillaries are ridd...	225	
23270	What would happen if everyone on Earth screame...	If everyone on Earth screamed at the same time...	0	If everyone Earth screamed time , would probab...	If everyone on Earth screamed at the same time...	100	

3 rows × 23 columns

Next, we will Split the dataset into independent variables (x) and the dependent (target) variable (y)

In [ ]:
```
# split into target and independent variables

# select all columns except for the specified ones
```

```
x = df.loc[:, ~df.columns.isin(['question', 'answer', 'is_human', 'filtered_answer'

#select is_human as target
y = df['is_human'] # target
```

Additionally, we will also scale the x values using sci-kit's StandardScaler. This scales each x value so that they have the same scale and dimension, which prevents differently-scaled variables from introducing biases into the modelling.

In [ ]:
```
# initialise scaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

Next, split the data into testing (20%) and training (80%) sets using sklearn's train_test_split

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.2, r
```

Initialise the 3 classification algorithms.

In [ ]:
```
sgdc = SGDClassifier(loss = 'hinge')
lr = LogisticRegression()
rfc = RandomForestClassifier()
```

Train the 3 classifiers using the training data

In [ ]:
```
sgdc.fit(x_train, y_train)
lr.fit(x_train, y_train)
rfc.fit(x_train, y_train)
```

Out[ ]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

Generate a prediction using the 3 classifiers

In [ ]:
```
# generate prediction
sgdc_predict = sgdc.predict(x_test)
lr_predict = lr.predict(x_test)
rfc_predict = rfc.predict(x_test)
```

Also, we will generate each model's predicted probability of each class prediction for evaluation using sklearn's predict_proba method.

For the SGDC, this requires a slight more complicated method of using a CalibratedClassifierCV to generate the probabilities, as the predict_proba method is not available for SGDCs. We will do this by using the method suggested in Reddy [2019], and use a CalibratedClassifierCV to compute the SGDC's probabilities (this is as SGDC does not support predict_proba by default)

In [ ]:
```
# predict LR/RFC probabilities

lr_proba = lr.predict_proba(x_test)
lr_proba = lr_proba[:, 1] # keep only the probability of the positive class

rfc_proba = rfc.predict_proba(x_test)
rfc_proba = rfc_proba[:, 1] # keep only the probability of the positive class
```

In [ ]:
```
# predict SGDC probabilties

# initalise a calibrator for the SGDC, and indicate the model is already prefitted
calibrator = CalibratedClassifierCV(sgdc, cv='prefit')
# fit the calibrator
model = calibrator.fit(x_train, y_train)

#generate probabilities
sgdc_proba = model.predict_proba(x_test)
# keep only the probability of the positive class
sgdc_proba = sgdc_proba[:, 1]
```

3.3 Featureset Testing

We will also test each of the 5 featuresets individually to determine their effect on model performance.

In the interest of fairness, we will be using the same classification algorithm (a simple LR model) to test each feature set.

First, we will need to split the feature sets.

In [ ]:
```
# select all columns except for the specified ones
x = df.loc[:, ~df.columns.isin(['question', 'answer', 'is_human', 'filtered_answer'

#select is_human as target
y = df['is_human'] # target

# split into each featureset
basic = x[['word_count', 'sentence_count', 'char_count', 'avg_sentence_len', 'avg_w
sentiment = x[['polarity', 'negativity', 'neutrality', 'subjectivity']]
syntactic = x[['lexical_density', 'NE_count', 'coref_count', 'avg_coref_len']]
frequency = x[['zipfian_difference', 'unique_word_proportion', 'stopword_proportion
phrasal = x[['cliche_count', 'ancient_phrase_count']]
```

We will also need to scale the values of each feature set.

We will first define a function to perform this task.

In [ ]:
```
def scale_values(featureset):
    scaler = StandardScaler() #initialise scaler
    featureset_scaled = scaler.fit_transform(featureset) #scale values
    return featureset_scaled
```

and call it on each featureset

In [ ]:
```
basic_scaled = scale_values(basic)
sentiment_scaled = scale_values(sentiment)
syntactic_scaled = scale_values(syntactic)
frequency_scaled = scale_values(frequency)
phrasal_scaled = scale_values(phrasal)
```

We will also need to split each feature set into training and testing data.

In [ ]:
```
x_basic_train, x_basic_test, y_basic_train, y_basic_test = train_test_split(basic_s
x_sentiment_train, x_sentiment_test, y_sentiment_train, y_sentiment_test = train_te
x_syntactic_train, x_syntactic_test, y_syntactic_train, y_syntactic_test = train_te
x_frequency_train, x_frequency_test, y_frequency_train, y_frequency_test = train_te
x_phrasal_train, x_phrasal_test, y_phrasal_train, y_phrasal_test = train_test_split
```

Next, we will define a function to fit the model, and generate a prediction and probability for each feature set using the LR model.

In [ ]:
```
def fit_predict_proba(x_train, x_test, y_train):
    lr = LogisticRegression() #initialise model
    lr.fit(x_train, y_train) # fit model
    prediction = lr.predict(x_test) # generate prediction
    probability = lr.predict_proba(x_test) # generate probabilities
    probability = probability[:, 1] # keep only the probability of the positive cla
    return prediction, probability
```

In [ ]:
```
basic_predict, basic_proba = fit_predict_proba(x_basic_train, x_basic_test, y_basic
sentiment_predict, sentiment_proba = fit_predict_proba(x_sentiment_train, x_sentime
syntactic_predict, syntactic_proba = fit_predict_proba(x_syntactic_train, x_syntact
frequency_predict, frequency_proba = fit_predict_proba(x_frequency_train, x_frequen
phrasal_predict, phrasal_proba = fit_predict_proba(x_phrasal_train, x_phrasal_test,
```

4. Evaluation

First, we will import the required modules

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDispla
import matplotlib.pyplot as plt
import numpy as np
```

Next, we will define some helper functions that will be used to plot confusion matrices, calculate EER, calculate performance stats, and add numerical labels to bar charts.

In [ ]:
```
# plots a confusion matrix display
def plot_con_matrix(y_test, y_predict):
    # create confusion matrix
    con_matrix = confusion_matrix(y_test, y_predict)
    # create confusion matrix display and set labels
    cm_display = ConfusionMatrixDisplay(con_matrix, display_labels=['gpt', 'human']
    # plot and show confusion matrix
    return cm_display
```

```
# calculates FPR and TPR
def calc_fpr_tpr(y_test, y_predict):

    # create confusion matrix
    con_matrix = confusion_matrix(y_test, y_predict)
    false_pos_rate = con_matrix[0][1] / (con_matrix[0][0] + con_matrix[0][1]) #calc
    true_pos_rate = con_matrix[1][1] / (con_matrix[1][0] + con_matrix[1][1]) #calcu

    return false_pos_rate, true_pos_rate

# calculates EER
# method referenced from James S. [2017].
def calc_eer(y_test, y_prob):
    # generate roc curve, and fpr, tpr lists and the threshold
    fpr, tpr, threshold = roc_curve(y_test, y_prob, pos_label = 1)
    # compute false negative rate
    fnr = 1 - tpr
    # find EER
    eer = fpr[np.nanargmin(np.absolute((fnr - fpr)))]

    return eer

# calculates accuracy, roc, eer, precision, recall
def calc_performance_stats(y_test, y_predict, y_prob):
    accuracy = accuracy_score(y_test, y_predict) # calculate accuracy
    roc_score = roc_auc_score(y_test, y_predict) # calculate ROC
    eer = calc_eer(y_test, y_prob) # calculate EER
    precision = precision_score(y_test, y_predict) # calculate precision
    recall = recall_score(y_test, y_predict) # calculate recall

    return accuracy, roc_score, eer, precision, recall

# add value labels to bar charts
def add_val_labels(x, y, ax):
    for i in range(len(x)):
        ax.text(x[i], y[i]/3, f'{y[i]:.3f}', ha = 'center') # draw value at 30% hei
```

4.1 Model Evaluation

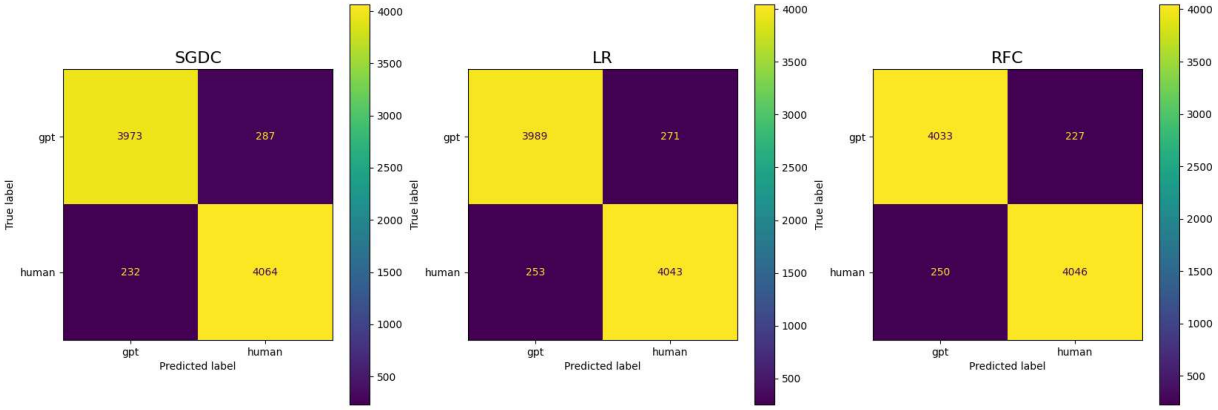
First, to get a brief visual overview of each model's performance, we will plot its confusion matrix and ROC curve.

First, we will plot the confusion matrix for each model.

In [ ]:
```
fig, axes = plt.subplots(1, 3, figsize = (20, 7))

axes[0].set_title("SGDC", size = 16) # set graph title
plot_con_matrix(y_test, sgdc_predict).plot(ax = axes[0]) # plot confusion matrix

axes[1].set_title("LR", size = 16)
plot_con_matrix(y_test, lr_predict).plot(ax = axes[1])
```

```
axes[2].set_title("RFC", size = 16)
plot_con_matrix(y_test, rfc_predict).plot(ax = axes[2])
```

Out[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x23161a94790>

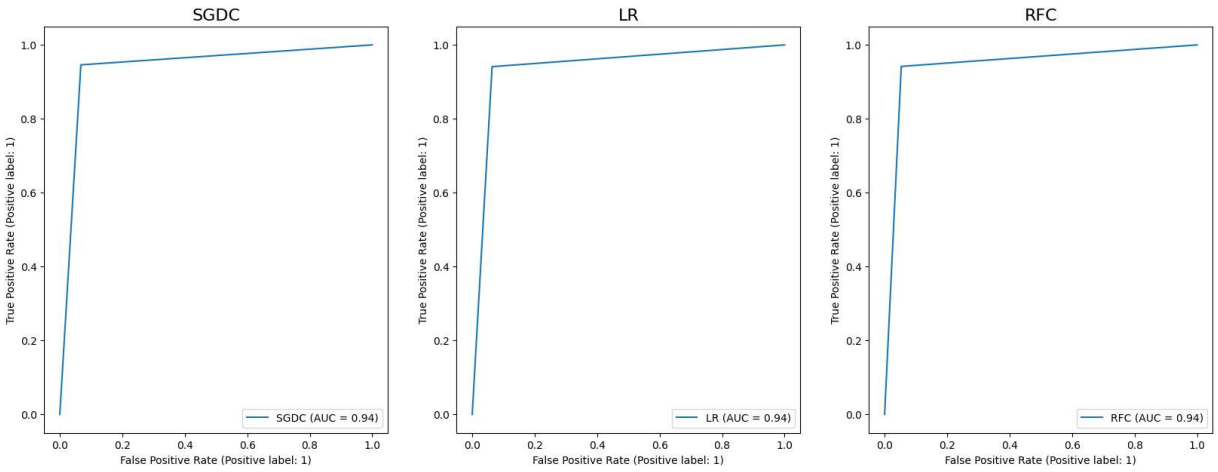


A brief look at each model's confusion matrix shows that all 3 models perform relatively similarly.

We will also plot each model's ROC curve.

In [ ]:
```
fig, axes = plt.subplots(1, 3, figsize = (20, 7))
axes[0].set_title("SGDC", size = 16) # set graph title
RocCurveDisplay.from_predictions(y_test, sgdc_predict, name='SGDC', ax = axes[0]) #

axes[1].set_title("LR", size = 16)
RocCurveDisplay.from_predictions(y_test, lr_predict, name = 'LR', ax = axes[1])

axes[2].set_title("RFC", size = 16)
RocCurveDisplay.from_predictions(y_test, rfc_predict, name = 'RFC', ax = axes[2])
```

Out[ ]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23162233bd0>



Once again, this shows that all 3 models do perform quite similarly, with high AUCROC scores. Of note, all 3 models seem to have a 'point of diminishing returns' around 0.935 TPR, where the TPR does not increase significantly against the increase in FPR.

Next, we will calculate each model's:

1. Accuracy
2. AUROC Score
3. Equal Error Rate (EER)
4. Precision
5. Recall
6. FPR
7. TPR

In [ ]:
```
# calculate acc, roc, eer, precision, recall
sgdc_acc, sgdc_roc, sgdc_eer, sgdc_precision, sgdc_recall = calc_performance_stats(
lr_acc, lr_roc, lr_eer, lr_precision, lr_recall = calc_performance_stats(y_test, lr
rfc_acc, rfc_roc, rfc_eer, rfc_precision, rfc_recall = calc_performance_stats(y_tes

# print out stats
print(f'SGDC - Accuracy: {sgdc_acc}, ROC: {sgdc_roc}, EER: {sgdc_eer}, Precision: {
print(f'LR -  Accuracy: {lr_acc}, ROC: {lr_roc}, EER: {lr_eer}, Precision: {lr_prec
print(f'RFC -  Accuracy: {rfc_acc}, ROC: {rfc_roc}, EER: {rfc_eer}, Precision: {rfc
```

```
SGDC - Accuracy: 0.9393408134642356, ROC: 0.9393126917932175, EER: 0.061267605633802
82, Precision: 0.9340381521489313, Recall: 0.9459962756052142
LR -  Accuracy: 0.9387564282374942, ROC: 0.9387464919873055, EER: 0.0612676056338028
2, Precision: 0.9371812702828002, Recall: 0.9411080074487895
RFC -  Accuracy: 0.9442496493688639, ROC: 0.944259973247305, EER: 0.0549295774647887
36, Precision: 0.9468757313362977, Recall: 0.9418063314711359
```

We will also visualise the performance of the model using some bar charts.

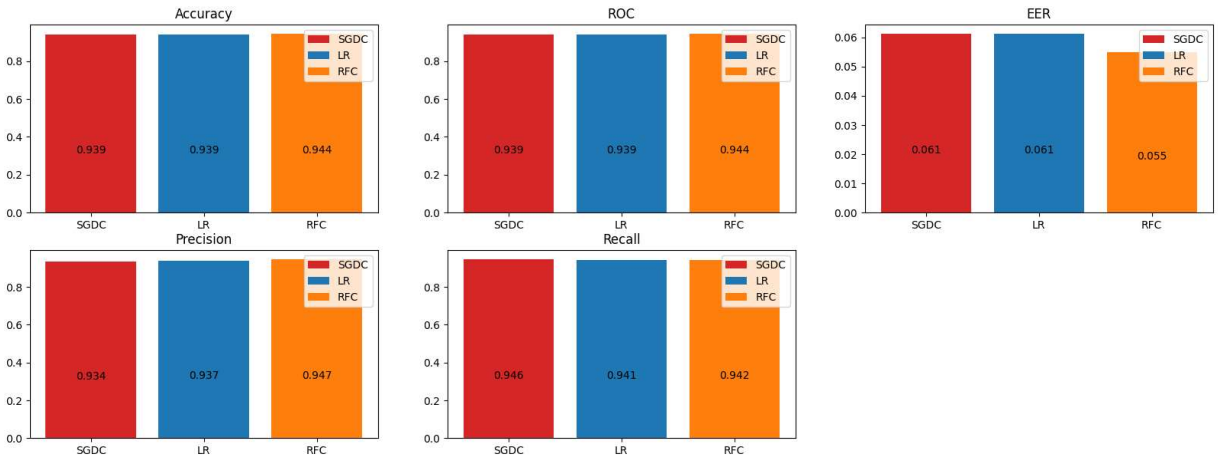
First, we will construct lists of the calculated stats, as well as some labels and colours

In [ ]:
```
# assemble list of stats
accuracies =  [sgdc_acc, lr_acc, rfc_acc]
rocs = [sgdc_roc, lr_roc, rfc_roc]
eers = [sgdc_eer, lr_eer, rfc_eer]
precisions = [sgdc_precision, lr_precision, rfc_precision]
recalls = [sgdc_recall, lr_recall, rfc_recall]

# also assemble list of labels and colours
labels = ['SGDC', 'LR', 'RFC']
colors = ['tab:red', 'tab:blue', 'tab:orange']
```

Finally, we will plot the bar charts

In [ ]:
```
fig, axes = plt.subplots(2, 3, figsize = (20, 7))

axes[0][0].set_title('Accuracy') # set plot title
axes[0][0].bar(labels, accuracies, label = labels, color = colors) # plot barplot
axes[0][0].legend() # add legend
add_val_labels(labels, accuracies, axes[0][0]) # add value labels

axes[0][1].set_title('ROC')
axes[0][1].bar(labels, rocs, label = labels, color = colors)
axes[0][1].legend()
add_val_labels(labels, rocs, axes[0][1])
```

```
axes[0][2].set_title('EER')
axes[0][2].bar(labels, eers, label = labels, color = colors)
axes[0][2].legend()
add_val_labels(labels, eers, axes[0][2])

axes[1][0].set_title('Precision')
axes[1][0].bar(labels, precisions, label = labels, color = colors)
axes[1][0].legend()
add_val_labels(labels, precisions, axes[1][0])

axes[1][1].set_title('Recall')
axes[1][1].bar(labels, recalls, label = labels, color = colors)
axes[1][1].legend()
add_val_labels(labels, recalls, axes[1][1])

fig.delaxes(axes[1][2]) # delete last empty plot
```



As can be observed from the bar charts, all 3 models perform similarly in terms of accuracy and ROC score. However, the RFC model performs significantly better in terms of EER (lower is better), and is slightly more precise than the other models.

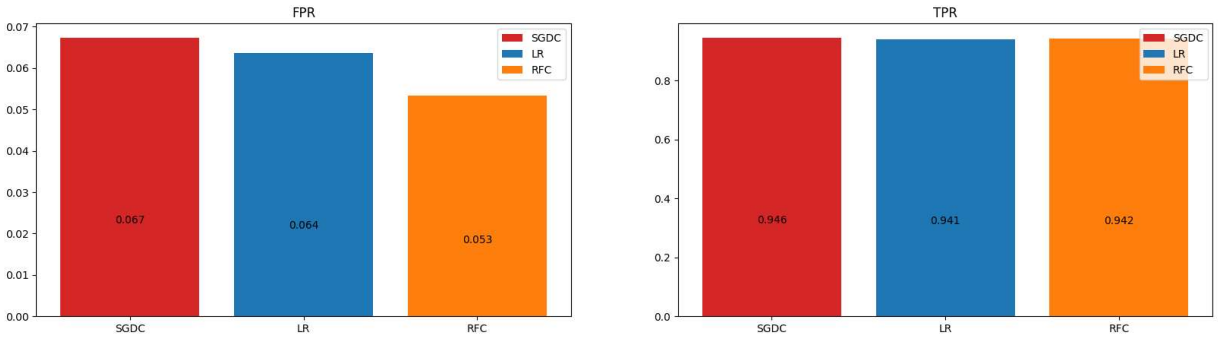
We will also compare the FPR and TPR of the models.

In [ ]:
```
sgdc_fpr, sgdc_tpr = calc_fpr_tpr(y_test, sgdc_predict)
lr_fpr, lr_tpr = calc_fpr_tpr(y_test, lr_predict)
rfc_fpr, rfc_tpr = calc_fpr_tpr(y_test, rfc_predict)

print(f'SGDC - FPR: {sgdc_fpr}, TPR: {sgdc_tpr}')
print(f'LR - FPR: {lr_fpr}, TPR: {lr_tpr}')
print(f'RFC - FPR: {rfc_fpr}, TPR: {rfc_tpr}')
```

```
SGDC - FPR: 0.06737089201877934, TPR: 0.9459962756052142
LR - FPR: 0.0636150234741784, TPR: 0.9411080074487895
RFC - FPR: 0.053286384976525825, TPR: 0.9418063314711359
```

In [ ]:
```
fprs = [sgdc_fpr, lr_fpr, rfc_fpr]
tprs = [sgdc_tpr, lr_tpr, rfc_tpr]
```

In [ ]:
```
fig, axes = plt.subplots(1, 2, figsize = (20, 5))

axes[0].set_title('FPR') # set plot title
axes[0].bar(labels, fprs, label = labels, color = colors) # plot barplot
axes[0].legend() # add legend
add_val_labels(labels, fprs, axes[0]) # add value labels

axes[1].set_title('TPR')
axes[1].bar(labels, tprs, label = labels, color = colors)
axes[1].legend()
add_val_labels(labels, tprs, axes[1])
```



As can be observed, the RFC has comparable TPR to the other models, but significantly lower FPR.

Since the RFC has significantly better EER and lower FPR, we would concluide that, contrary to published literature, the RFC model is the best performing model.

4.2 Featureset Evaluation

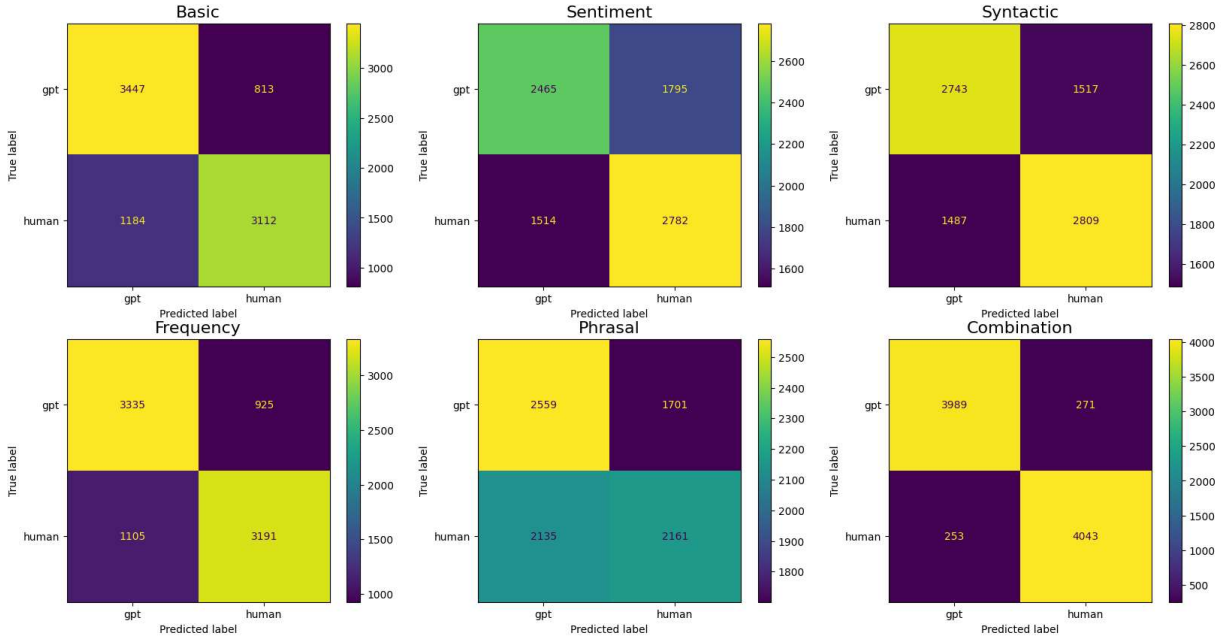
Now, we will evaluate the 5 featuresets using their trained LR models. Additionally, to compare each featureset's performance with the combined performance of all 5 featuresets, we will also be comparing the LR model from the previous model evaluation, which was trained on the entire combination of all 5 featuresets (called 'Combination')

Similarly, we will begin by obtaining a brief visual overview of each featureset's performance by the confusion matrix and ROC curve for each featureset.

In [ ]:
```
fig, axes = plt.subplots(2, 3, figsize = (20, 10))

axes[0][0].set_title("Basic", size = 16) # set graph title
plot_con_matrix(y_basic_test, basic_predict).plot(ax = axes[0][0]) # plot confusion

axes[0][1].set_title("Sentiment", size = 16)
plot_con_matrix(y_sentiment_test, sentiment_predict).plot(ax = axes[0][1])

axes[0][2].set_title("Syntactic", size = 16)
plot_con_matrix(y_syntactic_test, syntactic_predict).plot(ax = axes[0][2])

axes[1][0].set_title("Frequency", size = 16)
plot_con_matrix(y_frequency_test, frequency_predict).plot(ax = axes[1][0])
```

```
axes[1][1].set_title("Phrasal", size = 16)
plot_con_matrix(y_phrasal_test, phrasal_predict).plot(ax = axes[1][1])

axes[1][2].set_title("Combination", size = 16)
plot_con_matrix(y_test, lr_predict).plot(ax = axes[1][2])
```

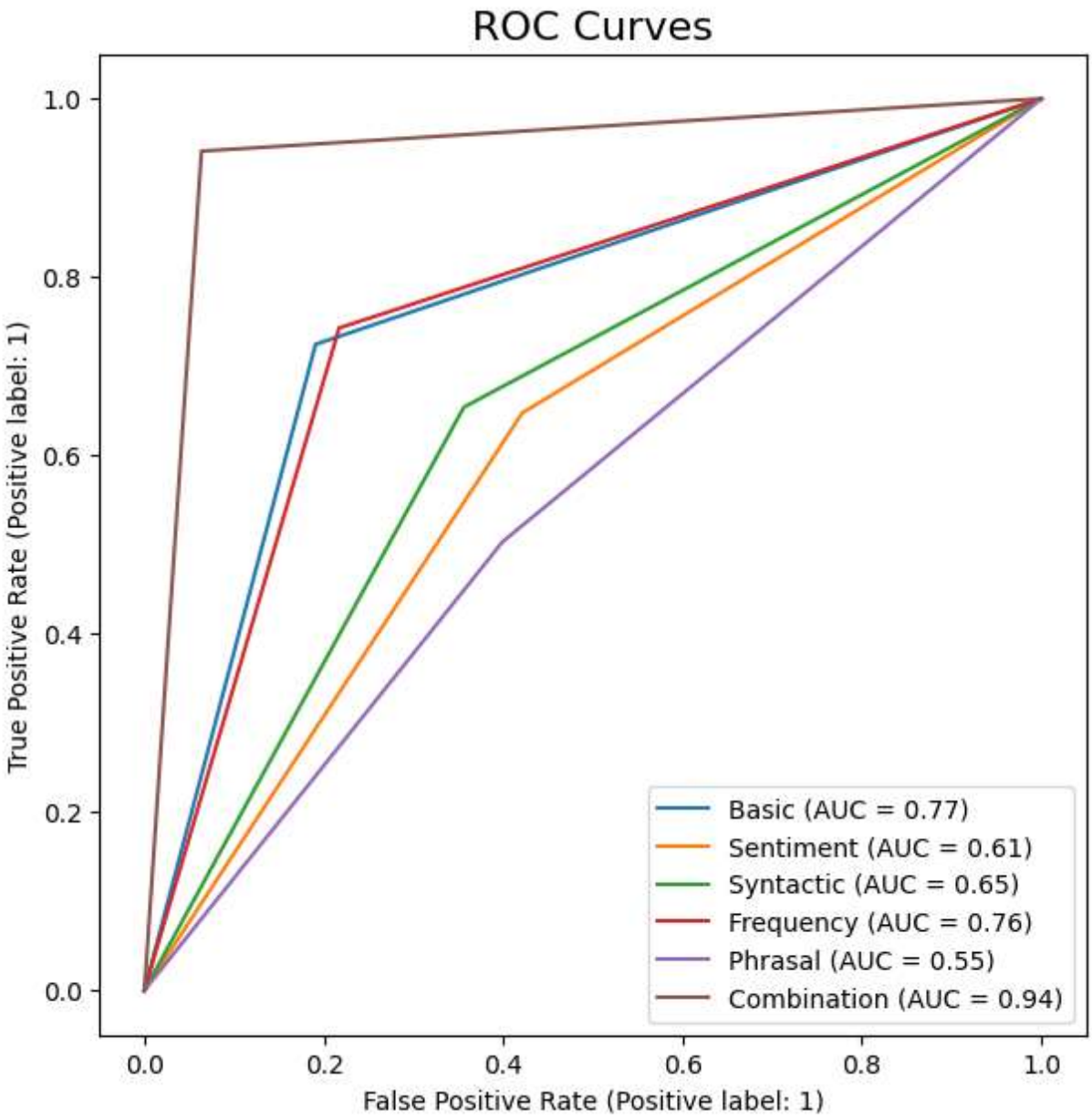
Out[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x231628d1b50>



The most obvious obervation from the confusion matrices are that the Basic and Frequency featuresets seem to yield significantly better performance than the other features, though they still perform considerably worse than Combination.

In [ ]:
```
fig, ax = plt.subplots(1,1, figsize = (7, 7))

ax.set_title("ROC Curves", size = 16) # set graph title

# plot curves
RocCurveDisplay.from_predictions(y_basic_test, basic_predict, name ='Basic', ax = a
RocCurveDisplay.from_predictions(y_sentiment_test, sentiment_predict, name ='Sentim
RocCurveDisplay.from_predictions(y_syntactic_test, syntactic_predict, name ='Syntac
RocCurveDisplay.from_predictions(y_frequency_test, frequency_predict, name ='Freque
RocCurveDisplay.from_predictions(y_phrasal_test, phrasal_predict, name ='Phrasal',
RocCurveDisplay.from_predictions(y_test, lr_predict, name ='Combination', ax = ax)
```

Out[ ]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23162bd9550>



Once again, this mirrors the results of the confusion matrices, with the Basic and Frequency featuresets significantly outperforming the rest, though falling short of Combination.

Next, we will again compare their performance statistics using bar charts.

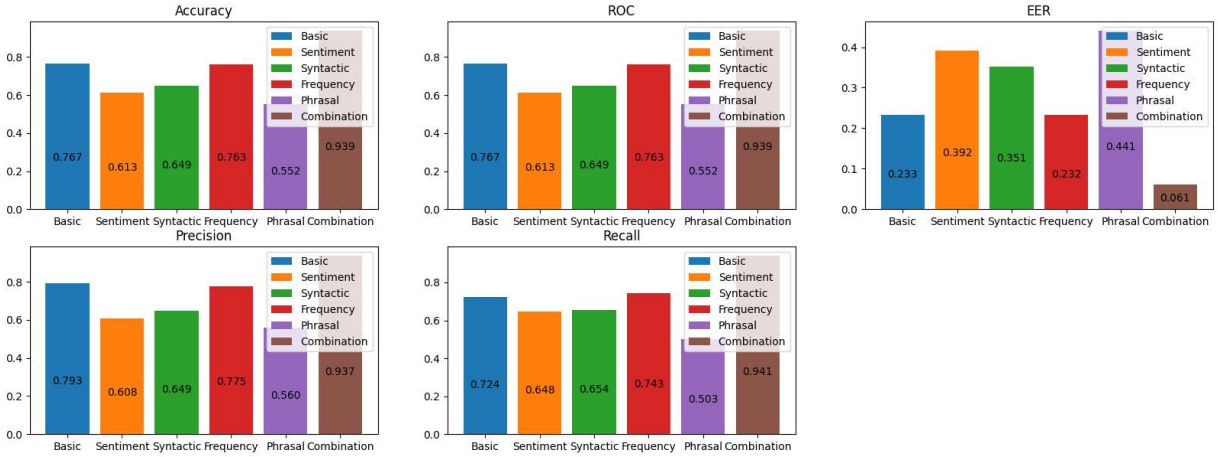
In [ ]:
```
# calculate acc, roc, err, precision, recall
basic_acc, basic_roc, basic_eer, basic_precision, basic_recall = calc_performance_s
sentiment_acc, sentiment_roc, sentiment_eer, sentiment_precision, sentiment_recall
syntactic_acc, syntactic_roc, syntactic_eer, syntactic_precision, syntactic_recall
frequency_acc, frequency_roc, frequency_eer, frequency_precision, frequency_recall
phrasal_acc, phrasal_roc, phrasal_eer, phrasal_precision, phrasal_recall = calc_per

# print stats
print(f'Basic - Accuracy: {basic_acc}, ROC: {basic_roc}, EER: {basic_eer}, Precisio
print(f'Sentiment - Accuracy: {sentiment_acc}, ROC: {sentiment_roc}, EER: {sentimen
print(f'Syntactic - Accuracy: {syntactic_acc}, ROC: {syntactic_roc}, EER: {syntacti
print(f'Frequency - Accuracy: {frequency_acc}, ROC: {frequency_roc}, EER: {frequenc
```

```
print(f'Phrasal - Accuracy: {phrasal_acc}, ROC: {phrasal_roc}, EER: {phrasal_eer},
print(f'Combination -  Accuracy: {lr_acc}, ROC: {lr_roc}, EER: {lr_eer}, Precision:
```

```
Basic - Accuracy: 0.7665965404394577, ROC: 0.7667748577123823, EER: 0.23309859154929
577, Precision: 0.7928662420382165, Recall: 0.7243947858472998
Sentiment - Accuracy: 0.6132538569424965, ROC: 0.6131088205208908, EER: 0.3915492957
7464787, Precision: 0.6078217172820625, Recall: 0.6475791433891993
Syntactic - Accuracy: 0.648901355773726, ROC: 0.6488803866026701, EER: 0.35140845070
422533, Precision: 0.6493296347665279, Recall: 0.6538640595903166
Frequency - Accuracy: 0.762739597942964, ROC: 0.7628239174338395, EER: 0.23215962441
314553, Precision: 0.7752672497570456, Recall: 0.7427839851024208
Phrasal - Accuracy: 0.5516596540439458, ROC: 0.5518651480578068, EER: 0.440845070422
53523, Precision: 0.5595546349041948, Recall: 0.5030260707635009
Combination -  Accuracy: 0.9387564282374942, ROC: 0.9387464919873055, EER: 0.0612676
0563380282, Precision: 0.9371812702828002, Recall: 0.9411080074487895
```

In [ ]:
```
# assemble list of stats
featuresets_accuracies =  [basic_acc, sentiment_acc, syntactic_acc, frequency_acc,
featuresets_rocs = [basic_roc, sentiment_roc, syntactic_roc, frequency_roc, phrasal
featuresets_eers = [basic_eer, sentiment_eer, syntactic_eer, frequency_eer, phrasal
featuresets_precisions = [basic_precision, sentiment_precision, syntactic_precision
featuresets_recalls = [basic_recall, sentiment_recall, syntactic_recall, frequency_

# also assemble list of labels and colours
featuresets_labels = ['Basic', 'Sentiment', 'Syntactic', 'Frequency', 'Phrasal', 'C
featuresets_colors = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple
```

In [ ]:
```
fig, axes = plt.subplots(2, 3, figsize = (20, 7))

axes[0][0].set_title('Accuracy') # add title
axes[0][0].bar(featuresets_labels, featuresets_accuracies, label = featuresets_labe
axes[0][0].legend() # add legend
add_val_labels(featuresets_labels, featuresets_accuracies, axes[0][0]) # ad value l

axes[0][1].set_title('ROC')
axes[0][1].bar(featuresets_labels, featuresets_rocs, label = featuresets_labels, co
axes[0][1].legend()
add_val_labels(featuresets_labels, featuresets_rocs, axes[0][1])

axes[0][2].set_title('EER')
axes[0][2].bar(featuresets_labels, featuresets_eers, label = featuresets_labels, co
axes[0][2].legend()
add_val_labels(featuresets_labels, featuresets_eers, axes[0][2])

axes[1][0].set_title('Precision')
axes[1][0].bar(featuresets_labels, featuresets_precisions, label = featuresets_labe
axes[1][0].legend()
add_val_labels(featuresets_labels, featuresets_precisions, axes[1][0])

axes[1][1].set_title('Recall')
axes[1][1].bar(featuresets_labels, featuresets_recalls, label = featuresets_labels,
axes[1][1].legend()
add_val_labels(featuresets_labels, featuresets_recalls, axes[1][1])

fig.delaxes(axes[1][2]) # delete empty plot
```

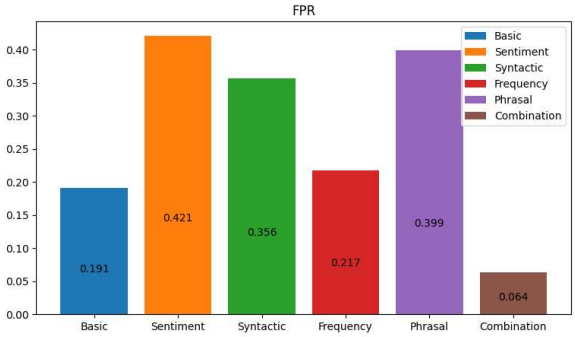


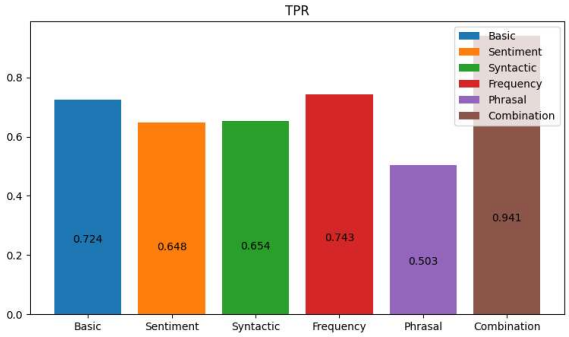
In [ ]:
```
basic_fpr, basic_tpr = calc_fpr_tpr(y_basic_test, basic_predict)
sentiment_fpr, sentiment_tpr = calc_fpr_tpr(y_sentiment_test, sentiment_predict)
syntactic_fpr, syntactic_tpr = calc_fpr_tpr(y_syntactic_test, syntactic_predict)
frequency_fpr, frequency_tpr = calc_fpr_tpr(y_frequency_test, frequency_predict)
phrasal_fpr, phrasal_tpr = calc_fpr_tpr(y_phrasal_test, phrasal_predict)

print(f'Basic- FPR: {basic_fpr}, TPR: {basic_tpr}')
print(f'Sentiment - FPR: {sentiment_fpr}, TPR: {sentiment_tpr}')
print(f'Syntactic - FPR: {syntactic_fpr}, TPR: {syntactic_tpr}')
print(f'Frequency - FPR: {frequency_fpr}, TPR: {frequency_tpr}')
print(f'Phrasal - FPR: {phrasal_fpr}, TPR: {phrasal_tpr}')
print(f'Combination - FPR: {lr_fpr}, TPR: {lr_tpr}')
```

```
Basic- FPR: 0.1908450704225352, TPR: 0.7243947858472998
Sentiment - FPR: 0.4213615023474178, TPR: 0.6475791433891993
Syntactic - FPR: 0.3561032863849765, TPR: 0.6538640595903166
Frequency - FPR: 0.21713615023474178, TPR: 0.7427839851024208
Phrasal - FPR: 0.39929577464788735, TPR: 0.5030260707635009
Combination - FPR: 0.0636150234741784, TPR: 0.9411080074487895
```

In [ ]:
```
fprs = [basic_fpr, sentiment_fpr, syntactic_fpr, frequency_fpr, phrasal_fpr, lr_fpr
tprs = [basic_tpr, sentiment_tpr, syntactic_tpr, frequency_tpr, phrasal_tpr, lr_tpr
```

In [ ]:
```
fig, axes = plt.subplots(1, 2, figsize = (20, 5))

axes[0].set_title('FPR') # set plot title
axes[0].bar(featuresets_labels, fprs, label = featuresets_labels, color = featurese
axes[0].legend() # add legend
add_val_labels(featuresets_labels, fprs, axes[0]) # add value labels

axes[1].set_title('TPR')
axes[1].bar(featuresets_labels, tprs, label = featuresets_labels, color = featurese
axes[1].legend()
add_val_labels(featuresets_labels, tprs, axes[1])
```





Once again, Basic and Frequency outperform the other featuresets by a large margin, though still falling far below Combination.

Of particular concern is the extremely high EER for each featureset compared to Combination. The nearest competitor to Combination (Frequency), is still nearly 4 times the EER of the Cbom. This is a strong indication that the combined model significantly outperforms individual featuresets, possibly beyond the previously observed amounts.

5. Additional RFC and BAF Featureset Combination Test

Now that we have identified the best performing model (RFC), and the best performing featuresets (Combination, or when isolated - Basic and Frequency), we would like to perform an additional test.

In this test, we would like to explore combining the two top performing isolated datasets (Basic and Frequency), and if such an RFC including only the Basic and Frequency datasets (called RFC(BAF)) would yield comparable/better performance than a combination RFC model (called RFC(Combination))

In [ ]:
```
#select is_human as target
y = df['is_human'] # target

basic_and_freq = x[['word_count', 'sentence_count', 'char_count', 'avg_sentence_len
combi_rfc = df.loc[:, ~df.columns.isin(['question', 'answer', 'is_human', 'filtered
basic_and_freq
```

Out[ ]:

	word_count	sentence_count	char_count	avg_sentence_len	avg_word_len	zipfian_d
0	228	7	1258	32.571429	5.517544	
1	154	5	851	30.800000	5.525974	
2	146	4	698	36.500000	4.780822	
3	178	5	944	35.600000	5.303371	
4	179	3	973	59.666667	5.435754	
...	...	...	...	...	...	
42775	132	5	646	26.400000	4.893939	
42776	132	7	590	18.857143	4.469697	
42777	357	17	1827	21.000000	5.117647	
42778	214	8	1178	26.750000	5.504673	
42779	177	6	950	29.500000	5.367232	

42780 rows × 8 columns

Scale the x values

In [ ]:
```
basic_and_freq_scaled = scale_values(basic_and_freq)
combi_rfc_scaled = scale_values(combi_rfc)
```

Split data set in to training and testing sets

In [ ]:
```
x_baf_train, x_baf_test, y_baf_train, y_baf_test = train_test_split(basic_and_freq_
x_combi_train, x_combi_test, y_combi_train, y_combi_test = train_test_split(combi_r
```

Initialise RFCs

In [ ]:
```
rfc_combination = RandomForestClassifier()
rfc_baf = RandomForestClassifier()
```

Train RFCs on training data

In [ ]:
```
rfc_combination.fit(x_combi_train, y_combi_train)
rfc_baf.fit(x_baf_train, y_baf_train)
```

Out[ ]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

Generate predictions and class probabilities

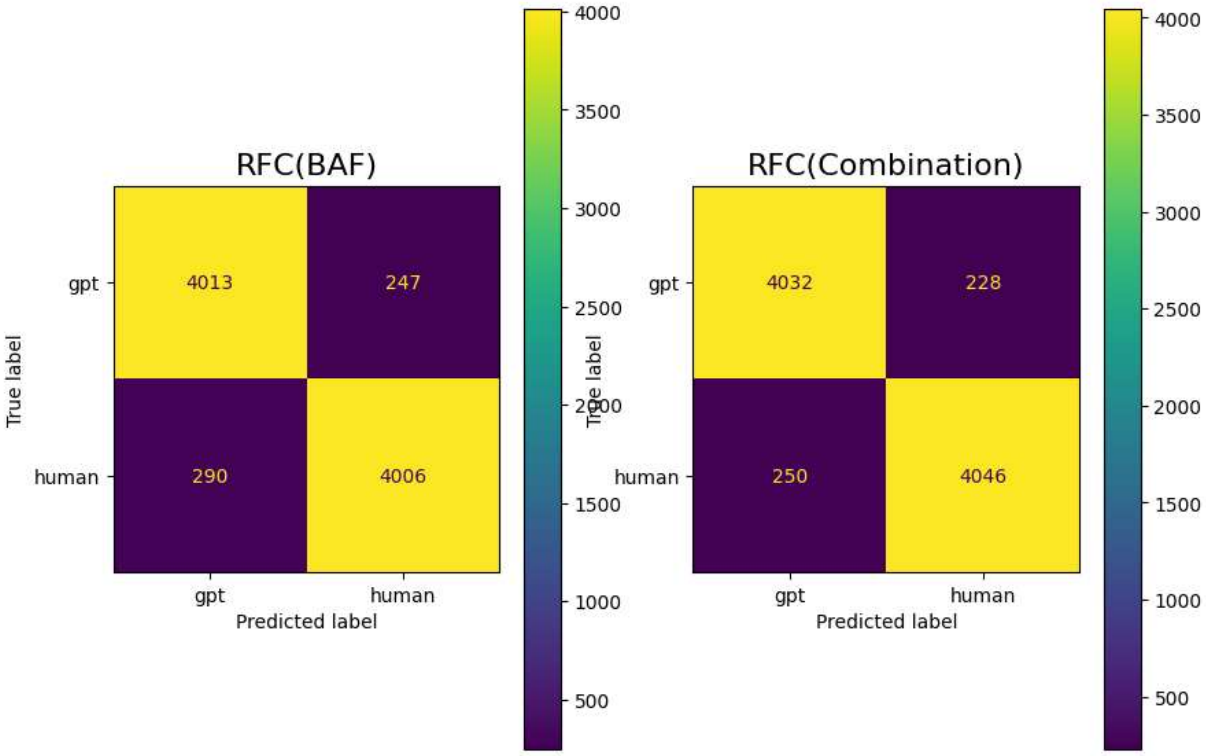
In [ ]:
```
rfc_combi_predict = rfc_combination.predict(x_combi_test)
rfc_baf_predict = rfc_baf.predict(x_baf_test)
```

In [ ]:
```
rfc_baf_proba = rfc_baf.predict_proba(x_baf_test)
rfc_baf_proba = rfc_baf_proba[:, 1] # keep only the probability of the positive cla

rfc_combi_proba = rfc_combination.predict_proba(x_combi_test)
rfc_combi_proba = rfc_combi_proba[:, 1] # keep only the probability of the positive
```

We will start by plotting the confusion matrices and ROC curves

In [ ]:
```
fig, ax = plt.subplots(1, 2, figsize = (10, 7))

ax[0].set_title("RFC(BAF)", size = 16) # set graph title
plot_con_matrix(y_baf_test, rfc_baf_predict).plot(ax = ax[0]) # plot confusion matr

ax[1].set_title("RFC(Combination)", size = 16)
plot_con_matrix(y_combi_test, rfc_combi_predict).plot(ax = ax[1])
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2316b18cc50>
```

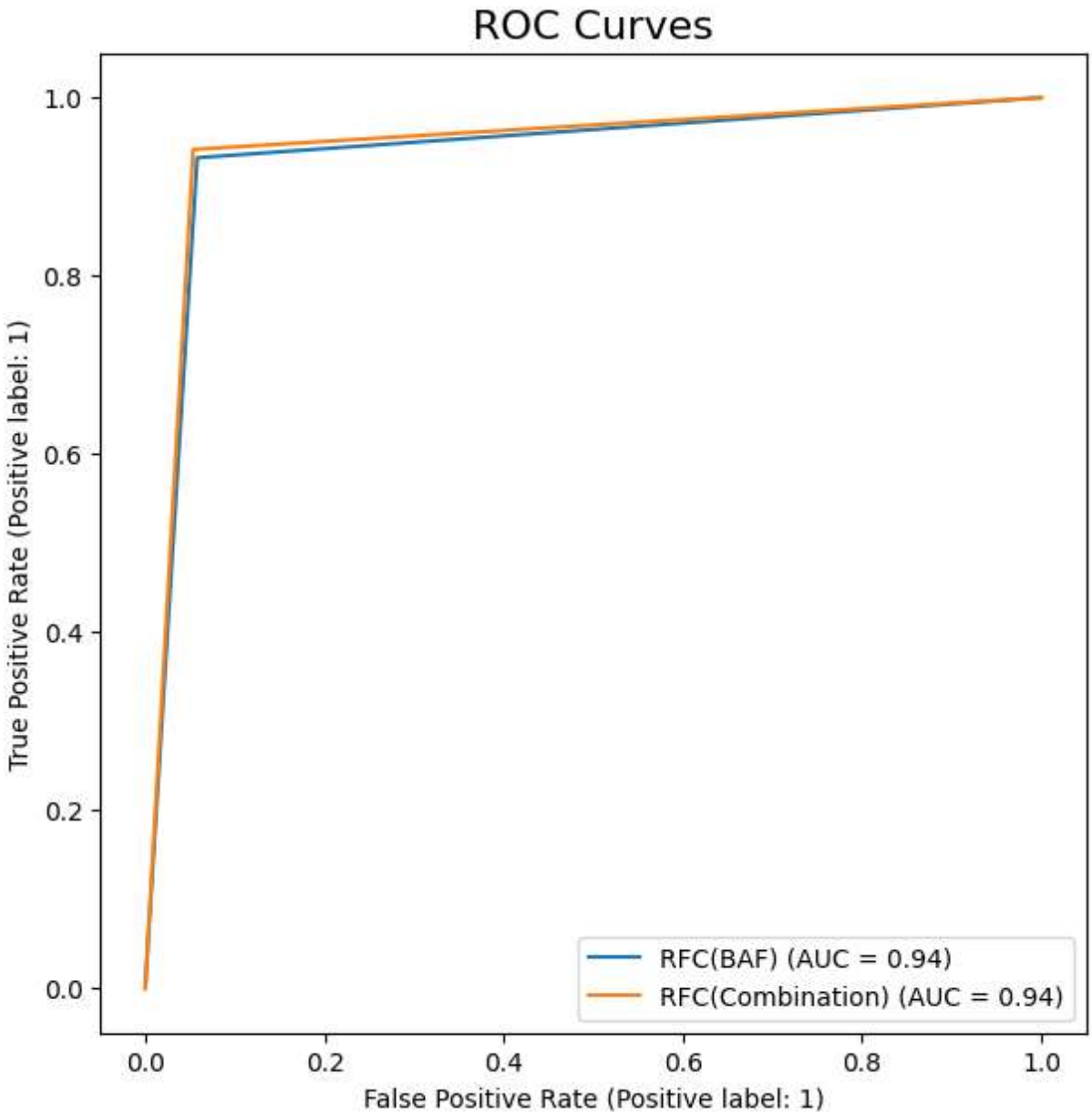


A cursory examination of both confusion matrices indicates that they have relatively similar performance.

In [ ]:
```
fig, ax = plt.subplots(1,1, figsize = (7, 7))

ax.set_title("ROC Curves", size = 16) # set graph title

# plot curves
```

```
RocCurveDisplay.from_predictions(y_baf_test, rfc_baf_predict, name ='RFC(BAF)', ax
RocCurveDisplay.from_predictions(y_combi_test, rfc_combi_predict, name ='RFC(Combin
```

Out[ ]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23163c6a450>



The ROC curves also reveal that both have very similar performance, though RFC(Combination) performs very slightly better.\

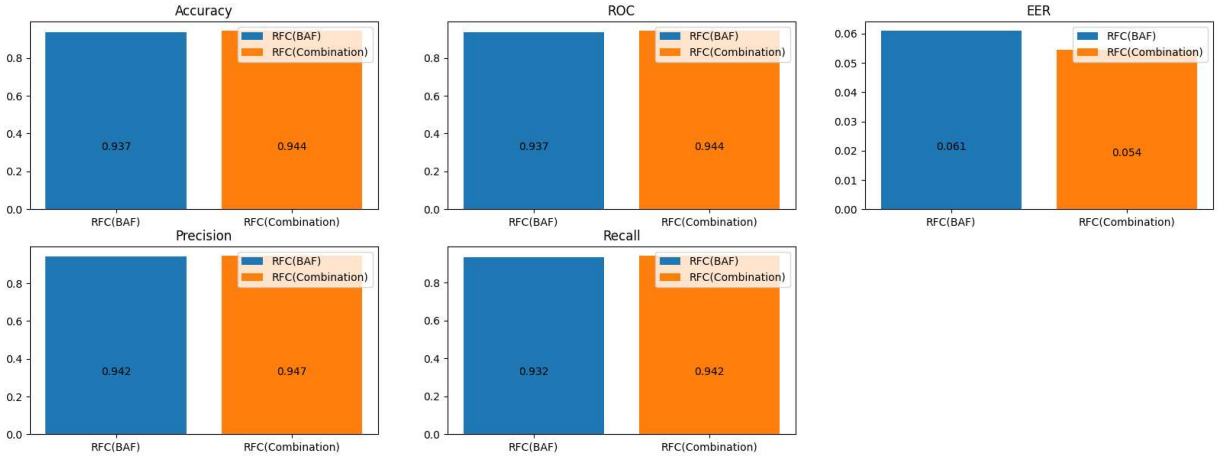
Next, we will compare the RFCs' performance statistics using bar charts

In [ ]:
```
# calculate performance stats
baf_acc, baf_roc, baf_eer, baf_precision, baf_recall = calc_performance_stats(y_baf
combi_acc, combi_roc, combi_eer, combi_precision, combi_recall = calc_performance_s


# print out stats
print(f'RFC(BAF) -  Accuracy: {baf_acc}, ROC: {baf_roc}, EER: {baf_eer}, Precision:
print(f'RFC(Combination) -  Accuracy: {combi_acc}, ROC: {combi_roc}, EER: {combi_ee
```

```
RFC(BAF) -  Accuracy: 0.9372370266479664, ROC: 0.9372570619246204, EER: 0.0610328638
4976526, Precision: 0.9419233482247825, Recall: 0.9324953445065177
RFC(Combination) -  Accuracy: 0.9441327723235157, ROC: 0.9441426023552864, EER: 0.05
4460093896713614, Precision: 0.9466541881141788, Recall: 0.9418063314711359
```

In [ ]:
```
# assemble combined list of stats
baf_combi_accuracies =  [baf_acc, combi_acc]
baf_combi_rocs = [baf_roc, combi_roc]
baf_combi_eers = [baf_eer, combi_eer]
baf_combi_precisions = [baf_precision, combi_precision]
baf_combi_recalls = [baf_recall, combi_recall]

# also assemble list of labels and colours
baf_combi_labels = ['RFC(BAF)', 'RFC(Combination)']
baf_combi_colors = ['tab:blue', 'tab:orange']
```

In [ ]:
```
fig, axes = plt.subplots(2, 3, figsize = (20, 7))

axes[0][0].set_title('Accuracy') # add title
axes[0][0].bar(baf_combi_labels, baf_combi_accuracies, label = baf_combi_labels, co
axes[0][0].legend() # add legend
add_val_labels(baf_combi_labels, baf_combi_accuracies, axes[0][0]) # ad value label

axes[0][1].set_title('ROC')
axes[0][1].bar(baf_combi_labels, baf_combi_rocs, label = baf_combi_labels, color =
axes[0][1].legend()
add_val_labels(baf_combi_labels, baf_combi_rocs, axes[0][1])

axes[0][2].set_title('EER')
axes[0][2].bar(baf_combi_labels, baf_combi_eers, label = baf_combi_labels, color =
axes[0][2].legend()
add_val_labels(baf_combi_labels, baf_combi_eers, axes[0][2])

axes[1][0].set_title('Precision')
axes[1][0].bar(baf_combi_labels, baf_combi_precisions, label = baf_combi_labels, co
axes[1][0].legend()
add_val_labels(baf_combi_labels, baf_combi_precisions, axes[1][0])

axes[1][1].set_title('Recall')
axes[1][1].bar(baf_combi_labels, baf_combi_recalls, label = baf_combi_labels, color
axes[1][1].legend()
add_val_labels(baf_combi_labels, baf_combi_recalls, axes[1][1])

fig.delaxes(axes[1][2]) # delete last empty plot
```



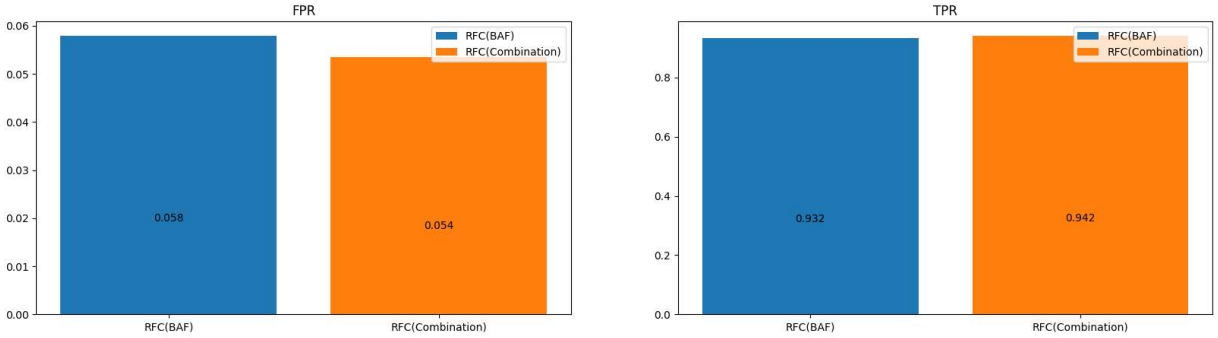
We will also compare their FPRs and TPRs

In [ ]:
```
baf_fpr, baf_tpr = calc_fpr_tpr(y_baf_test, rfc_baf_predict)
combi_fpr, combi_tpr = calc_fpr_tpr(y_combi_test, rfc_combi_predict)

print(f'BAF FPR: {baf_fpr}, TPR: {baf_tpr}')
print(f'Combination - FPR: {combi_fpr}, TPR: {combi_tpr}')
```

```
BAF FPR: 0.057981220657277, TPR: 0.9324953445065177
Combination - FPR: 0.05352112676056338, TPR: 0.9418063314711359
```

In [ ]:
```
fprs = [baf_fpr, combi_fpr]
tprs = [baf_tpr, combi_tpr]
```

In [ ]:
```
fig, axes = plt.subplots(1, 2, figsize = (20, 5))

axes[0].set_title('FPR') # set plot title
axes[0].bar(baf_combi_labels, fprs, label = baf_combi_labels, color = baf_combi_col
axes[0].legend() # add legend
add_val_labels(baf_combi_labels, fprs, axes[0]) # add value labels

axes[1].set_title('TPR')
axes[1].bar(baf_combi_labels, tprs, label = baf_combi_labels, color = baf_combi_col
axes[1].legend()
add_val_labels(baf_combi_labels, tprs, axes[1])
```



From the performance statistics, we can observe that the performance of both RFC models is extremely similar, with the exception of EER, where RFC(Combination) still performs significantly better.

This raises the interesting possibility that RFC(BAF) might be a better approach in certain circumstances. In situations where access to computing power or space is limited, RFC(BAF) might be a better option, as it requires a smaller number of features. Additionally, the features used in the Basic and Frequency datasets are computationally faster, and require significantly less technical expertise to extract.

6. Conclusions

From the testing and evaluation carried out above, we have identified the 2 main points of focus for our final prototype:

1. A RFC Classifier will be used.
2. The entire combination of all 5 features will be used. (Though RFC(BAF) is promising, RFC(Combination) still yields better overall performance)

6. References

[1] Lavanya Reddy. 2019. Answer to "AttributeError: probability estimates are not available for loss='hinge'." Stack Overflow. Retrieved June 28, 2023 from https://stackoverflow.com/a/57789235

[2] Craig. 2017. Answer to "Best Fit Line on Log Log Scales in python 2.7." Stack Overflow. Retrieved August 18, 2023 from https://stackoverflow.com/a/43838500

[3] James S. 2017. Answer to "Equal Error Rate in Python." Stack Overflow. Retrieved June 7, 2023 from https://stackoverflow.com/a/46026962