Imports

```
In [65]: import pandas as pd
         import numpy as np
         from sklearn.preprocessing import LabelEncoder
         from sklearn.model_selection import train_test_split
         import nltk
         from nltk.tokenize import word_tokenize
         from tqdm.notebook import tqdm, trange
         from termcolor import colored
         import fuzzywuzzy
         from fuzzywuzzy import fuzz
         from fuzzywuzzy import process
         # nltk.download()
```

Read in Data

```
In [66]: df = pd.read_csv("data/train.csv")
         df.head()
```

Out[66]:

	id	anchor	target	context	score
0	37d61fd2272659b1	abatement	abatement of pollution	A47	0.50
1	7b9652b17b68b7a4	abatement	act of abating	A47	0.75
2	36d72442aefd8232	abatement	active catalyst	A47	0.25
3	5296b0c19e1ce60e	abatement	eliminating process	A47	0.50
4	54c1e3b9184cb5b6	abatement	forest region	A47	0.00

Exploratory Data Analysis

```
In [67]: df.describe()
```

Out[67]:

	score
count	36473.000000
mean	0.362062
std	0.258335
min	0.000000
25%	0.250000
50%	0.250000
75%	0.500000
max	1.000000

```
In [68]: df.isnull().any()
Out[68]: id         False
         anchor     False
         target     False
         context    False
         score      False
         dtype: bool
```

Anchor Column Exploratory Analysis

```
In [69]: print(f"Number of uniques values in ANCHOR column: {colored(df.anchor.nunique(), 'blue')}")
         Number of uniques values in ANCHOR column: 733
```

Amount of values with respect to the length of anchor

```
In [70]: df['anchor_len'] = df['anchor'].str.split().str.len()
         df.anchor_len.value_counts()
Out[70]: 2    24485
         3     7556
         1     3377
         4      856
         5      199
         Name: anchor_len, dtype: int64
```

Top 20 anchor values

```
In [71]: df.anchor.value_counts().head(20)
Out[71]: component composite coating         152
         sheet supply roller                 150
         source voltage                      140
         perfluoroalkyl group                136
         el display                          135
         average pore size                   133
         speed control means                 130
         parallel orientation                127
         displacement mechanism              124
         cathode layer                       118
         flow openings                       117
         electromagnetic radiation source    115
         overall weight                      114
         central flow path                   111
         elastic assembly                    109
         dimensional location                109
         hydrocarbyl substituted succinic    108
         conductor particles                 108
         oxidizing enzyme                    107
         stabilizer members                  107
         Name: anchor, dtype: int64
```

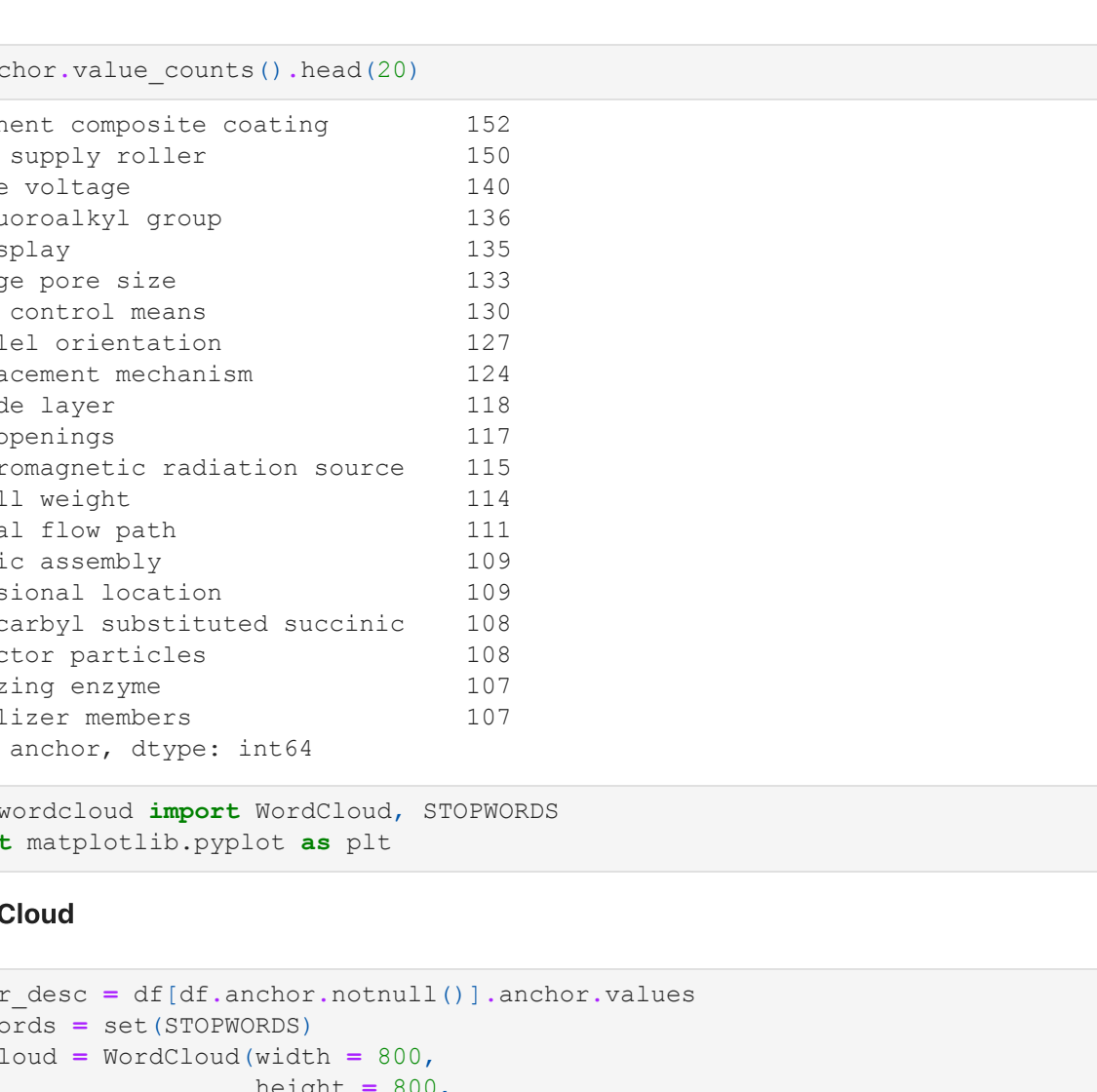
```
In [72]: from wordcloud import WordCloud, STOPWORDS
         import matplotlib.pyplot as plt
```

Word Cloud

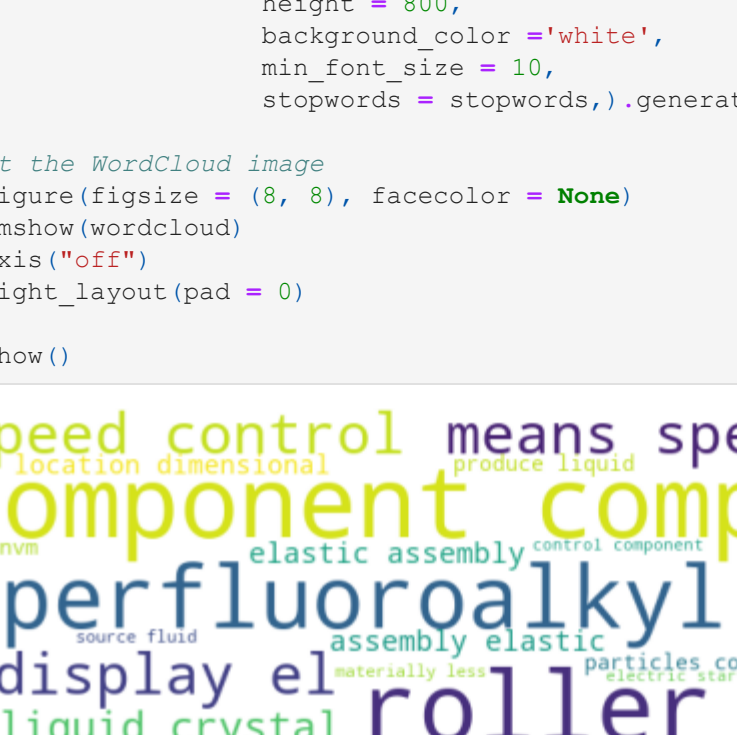
```
In [73]: anchor_desc = df[df.anchor.notnull()].anchor.values
         stopwords = set(STOPWORDS)
         wordcloud = WordCloud(width = 800,
                               height = 800,
                               background_color ='white',
                               min_font_size = 10,
                               stopwords = stopwords,).generate(' '.join(anchor_desc))

         # plot the WordCloud image
         plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)

         plt.show()
```




```
In [74]: df.anchor_len.hist(orientation='horizontal', color='#FFCF56')
Out[74]: <AxesSubplot:>
```



These are the unique amount of words in **anchor** column max is 5

```
In [75]: print("These are the possible amount of words in the anchor column: {}".format(np.unique(np.array(df['anchor'].
         These are the possible amount of words in the anchor column: [1 2 3 4 5]
```

Check for numerical values in the Anchor column

```
In [76]: pattern = '[0-9]'
         mask = df['anchor'].str.contains(pattern, na=False)
         df['num_target'] = mask
         df[mask]['anchor'].value_counts()
Out[76]: transistor m6               37
         her2 targeted               32
         vegfr2                      25
         conh2                       22
         dry coating composition1     2
         Name: anchor, dtype: int64
```

Target Column Exploratory Data Analysis

```
In [77]: print(f"Number of uniques values in TARGET column: {colored(df.target.nunique(), 'green')}")
         Number of uniques values in TARGET column: 29340
```

These are the unique amount of words in **target** column max is 15

```
In [78]: print("These are the possible amount of words in the target column: {}".format(np.unique(np.array(df['target'].
         These are the possible amount of words in the target column: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 15]
```

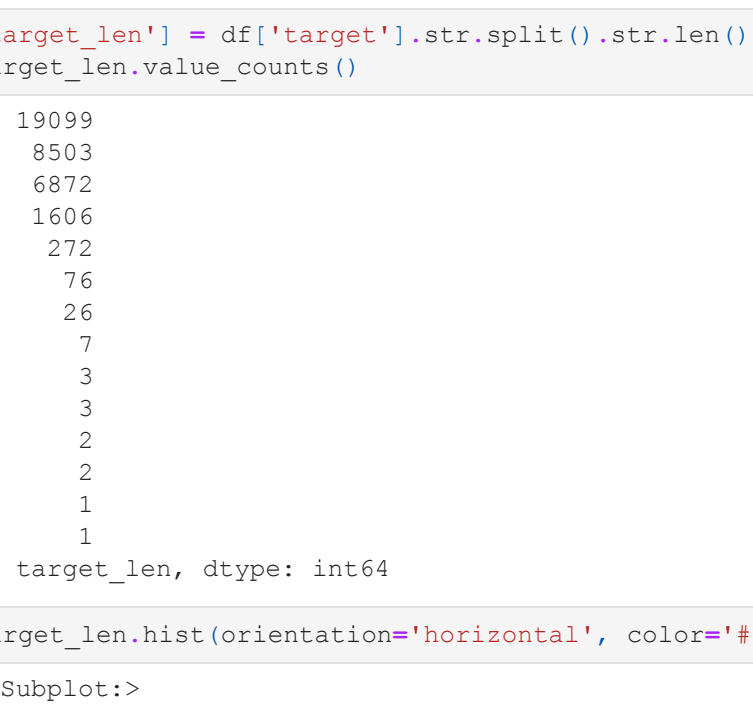
```
In [79]: target_desc = df[df.target.notnull()].target.values
         stopwords = set(STOPWORDS)
         wordcloud = WordCloud(width = 800,
                               height = 800,
                               background_color ='white',
                               min_font_size = 10,
                               stopwords = stopwords,).generate(' '.join(target_desc))

         # plot the WordCloud image
         plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)

         plt.show()
```


```
In [80]: df['target_len'] = df['target'].str.split().str.len()
         df.target_len.value_counts()
Out[80]: 2     19099
         3      8503
         1      6872
         4      1606
         5       272
         6        76
         7        26
         8         7
         10        3
         9         3
         13        2
         12        2
         15        1
         11        1
         Name: target_len, dtype: int64
```

```
In [81]: df.target_len.hist(orientation='horizontal', color='#FFCF56')
Out[81]: <AxesSubplot:>
```



Check for numerical values in Target

```
In [82]: pattern = '[0-9]'
         mask = df['target'].str.contains(pattern, na=False)
         df['num_target'] = mask
         df[mask]['target'].value_counts()
Out[82]: 1 bromopropane                   2
         consist of ethane 12 dioic acid  2
         tetrahydro 1h imidazo            2
         methyl 3h imidazo                2
         eastman g 3003                   2
                                         ..
         expandable 3 d scaffolding       1
         display 3 dimensional image      1
         sio2 fillers                     1
         3d position                      1
         h2o product                      1
         Name: target, Length: 112, dtype: int64
```

This is the amount of words in each row with respect to the column

- Must be applied to anchor and target

```
In [83]: np.unique(np.array(df['context']))
Out[83]: array(['A01', 'A21', 'A22', 'A23', 'A24', 'A41', 'A43', 'A44', 'A45',
                'A46', 'A47', 'A61', 'A62', 'A63', 'B01', 'B02', 'B03', 'B05',
                'B07', 'B08', 'B21', 'B22', 'B23', 'B24', 'B25', 'B27', 'B28',
                'B29', 'B31', 'B32', 'B41', 'B44', 'B60', 'B61', 'B62', 'B63',
                'B64', 'B65', 'B66', 'B67', 'B81', 'C01', 'C02', 'C03', 'C04',
                'C06', 'C07', 'C08', 'C09', 'C10', 'C11', 'C12', 'C13', 'C14',
                'C21', 'C22', 'C23', 'C25', 'D01', 'D03', 'D04', 'D05', 'D06',
                'D21', 'E01', 'E02', 'E03', 'E04', 'E05', 'E06', 'E21', 'F01',
                'F02', 'F03', 'F04', 'F15', 'F16', 'F17', 'F21', 'F22', 'F23',
                'F24', 'F25', 'F26', 'F27', 'F28', 'F41', 'F42', 'G01', 'G02',
                'G03', 'G04', 'G05', 'G06', 'G07', 'G08', 'G09', 'G10', 'G11',
                'G16', 'G21', 'H01', 'H02', 'H03', 'H04', 'H05'], dtype=object)
```

```
In [84]: df.context.value_counts().head(15)
Out[84]: H01    2186
         H04    2177
         G01    1812
         A61    1477
         F16    1091
         C07    1069
         G06    1063
         B60     916
         B01     891
         G02     877
         H03     798
         B65     764
         B41     650
         C08     649
         B23     648
         Name: context, dtype: int64
```

```
In [85]: print(np.unique(np.array(df['context'].apply(lambda x : len(x.split())))))
         [1]
```

```
In [86]: print(f"Number of uniques values in CONTEXT column: {colored(df.context.nunique(), 'yellow')}")
         Number of uniques values in CONTEXT column: 106
```

We need to seperate the context feature engineer the context

```
In [87]: df['section'] = df['context'].astype(str).str[0]
         df['classes'] = df['context'].astype(str).str[1:]
         df.head(10)
```

Out[87]:

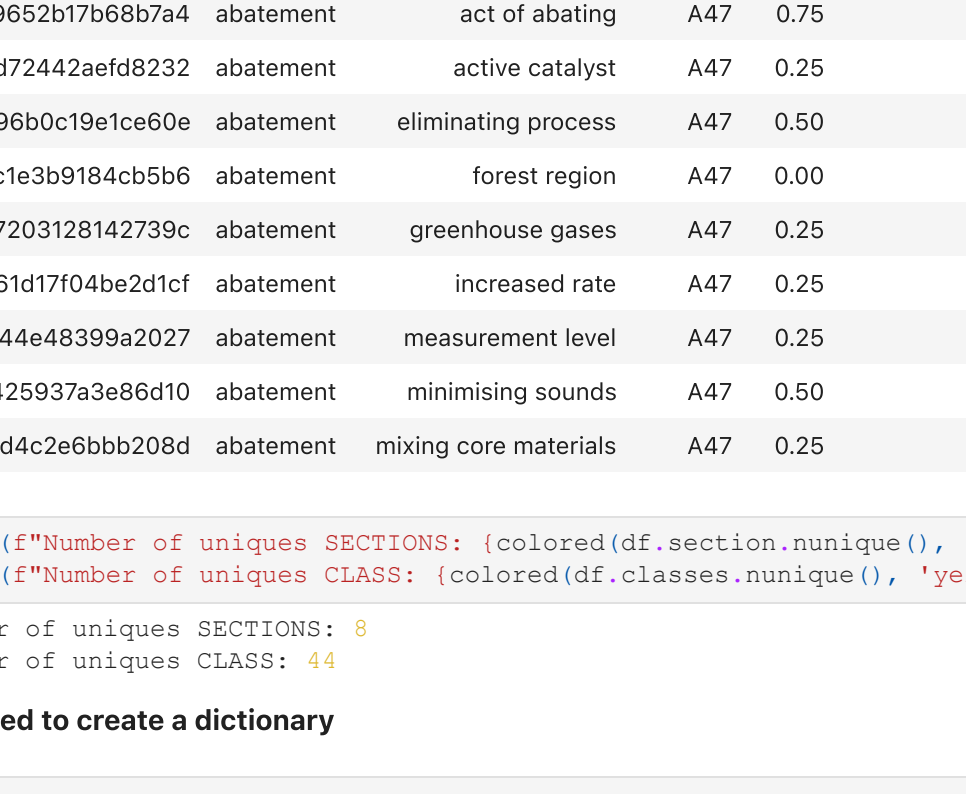
	id	anchor	target	context	score	anchor_len	num_target	target_len	section	classes
0	37d61fd2272659b1	abatement	abatement of pollution	A47	0.50	1	False	3	A	47
1	7b9652b17b68b7a4	abatement	act of abating	A47	0.75	1	False	3	A	47
2	36d72442aefd8232	abatement	active catalyst	A47	0.25	1	False	2	A	47
3	5296b0c19e1ce60e	abatement	eliminating process	A47	0.50	1	False	2	A	47
4	54c1e3b9184cb5b6	abatement	forest region	A47	0.00	1	False	2	A	47
5	067203128142739c	abatement	greenhouse gases	A47	0.25	1	False	2	A	47
6	061d17f04be2d1cf	abatement	increased rate	A47	0.25	1	False	2	A	47
7	e1f44e48399a2027	abatement	measurement level	A47	0.25	1	False	2	A	47
8	0a425937a3e86d10	abatement	minimising sounds	A47	0.50	1	False	2	A	47
9	ef2d4c2e6bbb208d	abatement	mixing core materials	A47	0.25	1	False	3	A	47

```
In [88]: print(f"Number of uniques SECTIONS: {colored(df.section.nunique(), 'yellow')}")
         print(f"Number of uniques CLASS: {colored(df.classes.nunique(), 'yellow')}")
         Number of uniques SECTIONS: 8
         Number of uniques CLASS: 44
```

We need to create a dictionary

```
In [89]: meanings = {"A" : "A - Human Necessities",
               "B" : "B - Operations and Transport",
               "C" : "C - Chemistry and Metallurgy",
               "D" : "D - Textiles",
               "E" : "E - Fixed Constructions",
               "F" : "F- Mechanical Engineering",
               "G" : "G - Physics",
               "H" : "H - Electricity",
               "Y" : "Y - Emerging Cross-Sectional Technologies"}
```

```
In [90]: df.replace({"section": meanings}).section.hist(orientation='horizontal', color='#FFCF56')
Out[90]: <AxesSubplot:>
```




```
In [91]: df.classes.value_counts().head(15)
Out[91]: 01    6851
         04    3630
         02    2310
         03    2130
         61    1771
         06    1468
         05    1325
         07    1294
         21    1269
         16    1145
         23    1126
         41     960
         60     916
         08     891
         09     872
         Name: classes, dtype: int64
```

Titles was gotten from doing more research on CPC Codes

```
In [92]: cpc_codes_df = pd.read_csv("data/cpc_codes_titles.csv", dtype=str)
         cpc_codes_df.head(10)
```

Out[92]:

	code	title	section	class	subclass	group	main_group
0	A	HUMAN NECESSITIES	A	NaN	NaN	NaN	NaN
1	A01	AGRICULTURE; FORESTRY; ANIMAL HUSBANDRY; HUNTI...	A	01	NaN	NaN	NaN
2	A01B	SOIL WORKING IN AGRICULTURE OR FORESTRY; PARTS...	A	01	B	NaN	NaN
3	A01B1/00	Hand tools (edge trimmers for lawns A01G3/06 ...	A	01	B	1	00
4	A01B1/02	Spades; Shovels {(hand-operated dredgers E02F3...	A	01	B	1	02
5	A01B1/022	{Collapsible; extensible; combinations with ot...	A	01	B	1	022
6	A01B1/024	{Foot protectors attached to the blade}	A	01	B	1	024
7	A01B1/026	{with auxiliary handles for facilitating lifting}	A	01	B	1	026
8	A01B1/028	{with ground abutment shoes or earth anchors f...	A	01	B	1	028
9	A01B1/04	with teeth	A	01	B	1	04

Let's join two datasets and add descriprion of context to our training DS

```
In [93]: df['context_desc'] = df['context'].map(cpc_codes_df.set_index('code')['title']).str.lower()
```

```
In [94]: df.head()
```

Out[94]:

	id	anchor	target	context	score	anchor_len	num_target	target_len	section	classes	context_desc
0	37d61fd2272659b1	abatement	abatement of pollution	A47	0.50	1	False	3	A	47	furniture; domestic articles or appliances; co...
1	7b9652b17b68b7a4	abatement	act of abating	A47	0.75	1	False	3	A	47	furniture; domestic articles or appliances; co...
2	36d72442aefd8232	abatement	active catalyst	A47	0.25	1	False	2	A	47	furniture; domestic articles or appliances; co...
3	5296b0c19e1ce60e	abatement	eliminating process	A47	0.50	1	False	2	A	47	furniture; domestic articles or appliances; co...
4	54c1e3b9184cb5b6	abatement	forest region	A47	0.00	1	False	2	A	47	furniture; domestic articles or appliances; co...

```
In [95]: df.drop("num_target",axis=1,inplace=True)
```

```
In [96]: df.head()
```

Out[96]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc
0	37d61fd2272659b1	abatement	abatement of pollution	A47	0.50	1	3	A	47	furniture; domestic articles or appliances; co...
1	7b9652b17b68b7a4	abatement	act of abating	A47	0.75	1	3	A	47	furniture; domestic articles or appliances; co...
2	36d72442aefd8232	abatement	active catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...
3	5296b0c19e1ce60e	abatement	eliminating process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...
4	54c1e3b9184cb5b6	abatement	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...

Tokenization

```
In [97]: df['target'] = df['target'].apply(lambda x : word_tokenize(x))
         df['anchor'] = df['anchor'].apply(lambda x : word_tokenize(x))
         df['context_desc'] = df['context_desc'].apply(lambda x : word_tokenize(x))
```

```
In [98]: df.head()
```

Out[98]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc
0	37d61fd2272659b1	[abatement]	[abatement, of, pollution]	A47	0.50	1	3	A	47	[furniture, ;, domestic, articles, or, applian...
1	7b9652b17b68b7a4	[abatement]	[act, of, abating]	A47	0.75	1	3	A	47	[furniture, ;, domestic, articles, or, applian...
2	36d72442aefd8232	[abatement]	[active, catalyst]	A47	0.25	1	2	A	47	[furniture, ;, domestic, articles, or, applian...
3	5296b0c19e1ce60e	[abatement]	[eliminating, process]	A47	0.50	1	2	A	47	[furniture, ;, domestic, articles, or, applian...
4	54c1e3b9184cb5b6	[abatement]	[forest, region]	A47	0.00	1	2	A	47	[furniture, ;, domestic, articles, or, applian...

Stopwords Removal

```
In [99]: from nltk.corpus import stopwords
         stop_words = set(stopwords.words('english'))
         df['target'] = df['target'].apply(lambda x : [w for w in x if not w in stop_words])
         df['anchor'] = df['anchor'].apply(lambda x : [w for w in x if not w in stop_words])
         df['context_desc'] = df['context_desc'].apply(lambda x : [w for w in x if not w in stop_words])
```

```
In [100]: df.head()
```

Out[100]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc
0	37d61fd2272659b1	[abatement]	[abatement, pollution]	A47	0.50	1	3	A	47	[furniture, ;, domestic, articles, appliances,...
1	7b9652b17b68b7a4	[abatement]	[act, abating]	A47	0.75	1	3	A	47	[furniture, ;, domestic, articles, appliances,...
2	36d72442aefd8232	[abatement]	[active, catalyst]	A47	0.25	1	2	A	47	[furniture, ;, domestic, articles, appliances,...
3	5296b0c19e1ce60e	[abatement]	[eliminating, process]	A47	0.50	1	2	A	47	[furniture, ;, domestic, articles, appliances,...
4	54c1e3b9184cb5b6	[abatement]	[forest, region]	A47	0.00	1	2	A	47	[furniture, ;, domestic, articles, appliances,...

Stemming

```
In [101]: from nltk.stem.snowball import SnowballStemmer
          snowBallStemmer = SnowballStemmer("english")
          df['target'] = df['target'].apply(lambda x : [snowBallStemmer.stem(word) for word in x])
          df['anchor'] = df['anchor'].apply(lambda x: [snowBallStemmer.stem(word) for word in x])
          df['context_desc'] = df['context_desc'].apply(lambda x: [snowBallStemmer.stem(word) for word in x])
```

```
In [102]: df.head()
```

Out[102]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc
0	37d61fd2272659b1	[abat]	[abat, pollut]	A47	0.50	1	3	A	47	[furnitur, ;, domest, articl, applianc, ;, cof...
1	7b9652b17b68b7a4	[abat]	[act, abat]	A47	0.75	1	3	A	47	[furnitur, ;, domest, articl, applianc, ;, cof...
2	36d72442aefd8232	[abat]	[activ, catalyst]	A47	0.25	1	2	A	47	[furnitur, ;, domest, articl, applianc, ;, cof...
3	5296b0c19e1ce60e	[abat]	[elimin, process]	A47	0.50	1	2	A	47	[furnitur, ;, domest, articl, applianc, ;, cof...
4	54c1e3b9184cb5b6	[abat]	[forest, region]	A47	0.00	1	2	A	47	[furnitur, ;, domest, articl, applianc, ;, cof...

Need to convert array to just strings

```
In [103]: df['anchor'] = df['anchor'].apply(lambda x : ' '.join(map(str,x)))
          df['target'] = df['target'].apply(lambda x : ' '.join(map(str,x)))
          df['context_desc'] = df['context_desc'].apply(lambda x : ' '.join(map(str,x)))
```

```
In [104]: df.head()
```

Out[104]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	3	A	47	furnitur ; domest articl applianc ; coffe mill...
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	3	A	47	furnitur ; domest articl applianc ; coffe mill...
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furnitur ; domest articl applianc ; coffe mill...
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furnitur ; domest articl applianc ; coffe mill...
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furnitur ; domest articl applianc ; coffe mill...

```
In [105]: from sentence_transformers import SentenceTransformer
          from sentence_transformers import SentenceTransformer, util
          model = SentenceTransformer('average_word_embeddings_komninos')
```

```
In [106]: def sent_trans(df,cpc_codes_df):

              pattern = '[0-9]'
              mask = df['anchor'].str.contains(pattern, na=False)
              df['num_anchor'] = mask
              mask = df['target'].str.contains(pattern, na=False)
              df['num_target'] = mask
              df['context_desc'] = df['context'].map(cpc_codes_df.set_index('code')['title']).str.lower()

              df['anchor_encoded'] = df.apply(lambda x: model.encode(x['anchor']),axis=1)
              df['target_encoded'] = df.apply(lambda x: model.encode(x['target']),axis=1)
              df['context_desc_encoded'] = df.apply(lambda x: model.encode(x['context_desc']),axis=1)

              df['score_at_sent'] = df.apply(lambda x: float(util.pytorch_cos_sim(x['anchor_encoded'], x['target_encoded']
              df['score_ac_sent'] = df.apply(lambda x: float(util.pytorch_cos_sim(x['anchor_encoded'], x['context_desc_en
              df['score_tc_sent'] = df.apply(lambda x: float(util.pytorch_cos_sim(x['target_encoded'], x['context_desc_en

              df['sent_c_score'] = df['score_ac_sent'] + df['score_tc_sent']
              df['sent_total'] = df['score_at_sent'] + df['sent_c_score']

              df['sent_avg_at']  = df['score_at_sent']/df['sent_total']
              df['sent_avg_c'] = df['sent_c_score']/df['sent_total']


              df.drop([ 'anchor_encoded','target_encoded','context_desc_encoded'], axis=1, inplace=True)


              return df
```

```
In [107]: df = sent_trans(df.copy(), cpc_codes_df)
          df.head()
```

Out[107]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	num_anchor	num_ta
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	3	A	47	furniture; domestic articles or appliances; co...	False	F
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	3	A	47	furniture; domestic articles or appliances; co...	False	F
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...	False	F
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	False	F
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...	False	F

```
In [108]: from bert_score import BERTScorer
          scorer = BERTScorer(lang="en", rescale_with_baseline=True)
          def bert_score(df):
              df['bert_at'] = df.apply(lambda x: float(scorer.score([x['anchor']],[x['target']])[2].mean()),axis=1)
              df['bert_ac'] = df.apply(lambda x: float(scorer.score([x['anchor']],[x['context_desc']])[2].mean()),axis=1)
              df['bert_tc'] = df.apply(lambda x: float(scorer.score([x['target']],[x['context_desc']])[2].mean()),axis=1)

              df['bert_c_score'] = df['bert_ac'] + df['bert_tc']
              df['bert_total'] = df['bert_at'] + df['bert_c_score']

              df['bert_avg_at']  = df['bert_at']/df['bert_total']
              df['bert_avg_c'] = df['bert_c_score']/df['bert_total']

              #df.drop([ 'bert_ac', 'bert_tc','bert_c_score','bert_at'], 1, inplace=True)
              return df
```

```
Some weights of the model checkpoint at roberta-large were not used when initializing RobertaModel: ['lm_head.d
ense.bias', 'lm_head.layer_norm.weight', 'lm_head.layer_norm.bias', 'lm_head.dense.weight', 'lm_head.decoder.we
ight', 'lm_head.bias']
- This IS expected if you are initializing RobertaModel from the checkpoint of a model trained on another task
or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining
model).
- This IS NOT expected if you are initializing RobertaModel from the checkpoint of a model that you expect to b
e exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification mo
del).
```

```
In [109]: df = bert_score(df.copy())
          df.head()
```

```
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty reference sentence detected; setting raw BERTScores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
```

Out[109]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	sent_total	sent_
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	3	A	47	furniture; domestic articles or appliances; co...	...	0.000000	
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	3	A	47	furniture; domestic articles or appliances; co...	...	0.232663	
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.356796	
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.321934	
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.366135	

5 rows × 26 columns

```
In [110]: import spacy
          nlp = spacy.load("en_core_web_lg")
          def spacy_score(df):
              df['spacy_at'] = df.apply(lambda x: nlp(str(x['anchor'])).similarity(nlp(str(x['target']))),axis=1)
              df['spacy_ac'] = df.apply(lambda x: nlp(str(x['anchor'])).similarity(nlp(str(x['context_desc']))),axis=1)
              df['spacy_tc'] = df.apply(lambda x: nlp(str(x['target'])).similarity(nlp(str(x['context_desc']))),axis=1)

              df['spacy_c_score'] = df['spacy_ac'] + df['spacy_tc']
              df['spacy_total'] = df['spacy_at'] + df['spacy_c_score']

              df['spacy_avg_at']  = df['spacy_at']/df['spacy_total']
              df['spacy_avg_c'] = df['spacy_c_score']/df['spacy_total']

              #df.drop([ 'spacy_ac', 'spacy_tc','spacy_at','spacy_c_score','spacy_total'], 1, inplace=True)
              return df
```

```
In [111]: df = spacy_score(df.copy())
          df.head()
```

```
C:\Users\Michael\AppData\Local\Temp\ipykernel_25672\3675883088.py:4: UserWarning: [W008] Evaluating Doc.similar
ity based on empty vectors.
  df['spacy_at'] = df.apply(lambda x: nlp(str(x['anchor'])).similarity(nlp(str(x['target']))),axis=1)
C:\Users\Michael\AppData\Local\Temp\ipykernel_25672\3675883088.py:5: UserWarning: [W008] Evaluating Doc.similar
ity based on empty vectors.
  df['spacy_ac'] = df.apply(lambda x: nlp(str(x['anchor'])).similarity(nlp(str(x['context_desc']))),axis=1)
C:\Users\Michael\AppData\Local\Temp\ipykernel_25672\3675883088.py:6: UserWarning: [W008] Evaluating Doc.similar
ity based on empty vectors.
  df['spacy_tc'] = df.apply(lambda x: nlp(str(x['target'])).similarity(nlp(str(x['context_desc']))),axis=1)
```

Out[111]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	bert_total	bert_
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	3	A	47	furniture; domestic articles or appliances; co...	...	-0.099302	-5.0
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	3	A	47	furniture; domestic articles or appliances; co...	...	-0.228251	-1.6
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...	...	-0.731640	0.1
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	-0.530172	-0.0
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...	...	-0.206242	-0.8

5 rows × 33 columns

Sets are important here because sets can only contain unique values

In [112…]:

```
def create_feature(df, cpc_codes_df):
    df['anchor_len'] = df['anchor'].apply(lambda x: len(x.split(' ')))
    df['target_len'] = df['target'].apply(lambda x: len(x.split(' ')))

    pattern = '[0-9]'
    mask = df['anchor'].str.contains(pattern, na=False)
    df['num_anchor'] = mask
    mask = df['target'].str.contains(pattern, na=False)
    df['num_target'] = mask

    df['context_desc'] = df['context'].map(cpc_codes_df.set_index('code')['title']).str.lower()

    fuzzy_anchor_target_scores = []
    fuzzy_anchor_context_scores = []
    fuzzy_taget_context_scores = []
    for index, row in df.iterrows():
        fuzzy_anchor_target_scores.append(fuzz.ratio(row['anchor'], row['target']))
        fuzzy_anchor_context_scores.append(fuzz.ratio(row['anchor'], row['context_desc']))
        fuzzy_taget_context_scores.append(fuzz.ratio(row['context_desc'], row['target']))
    df['fuzzy_at_score'] = fuzzy_anchor_target_scores
    df['fuzzy_ac_score'] = fuzzy_anchor_context_scores
    df['fuzzy_tc_score'] = fuzzy_taget_context_scores

    df['fuzzy_c_score'] = df['fuzzy_ac_score'] + df['fuzzy_tc_score']

    df['fuzzy_total'] = df['fuzzy_at_score'] + df['fuzzy_c_score']

    df['fuzzy_avg_at']  = df['fuzzy_at_score']/df['fuzzy_total']
    df['fuzzy_avg_c'] = df['fuzzy_c_score']/df['fuzzy_total']



    #df.drop([ 'fuzzy_ac_score', 'fuzzy_tc_score'], 1, inplace=True)

    return df
```

In [113…]:

```
df = create_feature(df.copy(), cpc_codes_df)
df.head()
```

Out[113]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	spacy_total	spac
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.000000	
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.388094	
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.085102	
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.427271	
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.415349	

5 rows × 40 columns

In [114…]:

```
all_words = set()
```

This functions are not meant to return anything but to get the words into the set very quickly using a lambda function

In [115…]:

```
def separator_all_words(arr):
        all_words.add(arr)
```

In [116…]:

```
df.head()
```

Out[116]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	spacy_total	spac
0	37d61fd2272659b1	abat	abat pollut	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.000000	
1	7b9652b17b68b7a4	abat	act abat	A47	0.75	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.388094	
2	36d72442aefd8232	abat	activ catalyst	A47	0.25	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.085102	
3	5296b0c19e1ce60e	abat	elimin process	A47	0.50	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.427271	
4	54c1e3b9184cb5b6	abat	forest region	A47	0.00	1	2	A	47	furniture; domestic articles or appliances; co...	...	0.415349	

5 rows × 40 columns

In [117…]:

```
df[df.columns[15:]]
```

Out[117]:

	sent_c_score	sent_total	sent_avg_at	sent_avg_c	bert_at	bert_ac	bert_tc	bert_c_score	bert_total	bert_avg_at
0	0.000000	0.000000	NaN	NaN	0.502910	-0.295080	-0.307132	-0.602212	-0.099302	-5.064444
1	0.232663	0.232663	0.000000	1.000000	0.366209	-0.295080	-0.299380	-0.594460	-0.228251	-1.604412
2	0.356796	0.356796	0.000000	1.000000	-0.084905	-0.295080	-0.351655	-0.646735	-0.731640	0.116048
3	0.321934	0.321934	0.000000	1.000000	0.019158	-0.295080	-0.254250	-0.549330	-0.530172	-0.036136
4	0.366135	0.366135	0.000000	1.000000	0.172460	-0.295080	-0.083622	-0.378702	-0.206242	-0.836199
...	...	...	...	...	...	...	...	...	...	...
36468	0.793261	1.321759	0.399844	0.600156	0.815021	0.068167	0.037802	0.105969	0.920990	0.884940
36469	0.769052	1.301905	0.409287	0.590713	0.291548	0.068167	0.067315	0.135482	0.427030	0.682733
36470	0.725379	1.219451	0.405160	0.594840	0.502922	0.068167	-0.000069	0.068098	0.571020	0.880744
36471	0.793261	1.321759	0.399844	0.600156	0.234150	0.068167	-0.012308	0.055859	0.290009	0.807389
36472	0.793261	1.321759	0.399844	0.600156	0.179098	0.068167	-0.044824	0.023343	0.202441	0.884691

36473 rows × 25 columns

In [118…]:

```
df['anchor'].apply(lambda x: separator_all_words(x))
df['target'].apply(lambda x: separator_all_words(x))
df['context_desc'].apply(lambda x: separator_all_words(x))
df['section'].apply(lambda x: separator_all_words(x))
# df['context'].apply(lambda x: separator_all_words(x))
```

Out[118]:

```
0        None
1        None
2        None
3        None
4        None
         ...
36468    None
36469    None
36470    None
36471    None
36472    None
Name: section, Length: 36473, dtype: object
```

List comprehension of converting a set to a list

In [119…]:

```
entire = [i for i in all_words]
```

Label Encoding

In [120…]:

```
labelencoder = LabelEncoder()
labelencoder.fit(entire)
```

Out[120]:

```
LabelEncoder()
```

In [121…]:

```
df.to_csv('without_encoding_data_file.csv',index=None)
```

For loop runs a lot faster than the lambda function idk why

In [122…]:

```
for i in tqdm(np.unique(df['anchor'])):
    df['anchor'].replace(i,labelencoder.transform([i])[0],inplace=True)
```

```
  0%|          | 0/733 [00:00<?, ?it/s]
```

In [123…]:

```
for i in tqdm(np.unique(df['target'])):
    df['target'].replace(i,labelencoder.transform([i])[0],inplace=True)
```

```
  0%|          | 0/26589 [00:00<?, ?it/s]
```

In [124…]:

```
for i in tqdm(np.unique(df['section'])):
    df['section'].replace(i,labelencoder.transform([i])[0],inplace=True)
```

```
  0%|          | 0/8 [00:00<?, ?it/s]
```

In [125…]:

```
for i in tqdm(np.unique(df['context_desc'])):
    df['context_desc'].replace(i,labelencoder.transform([i])[0],inplace=True)
```

```
  0%|          | 0/106 [00:00<?, ?it/s]
```

In [126…]:

```
df.head()
```

Out[126]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	spacy_total	spacy
0	37d61fd2272659b1	44	46	A47	0.50	1	2	36	47	10103	...	0.000000	
1	7b9652b17b68b7a4	44	257	A47	0.75	1	2	36	47	10103	...	0.388094	
2	36d72442aefd8232	44	276	A47	0.25	1	2	36	47	10103	...	0.085102	
3	5296b0c19e1ce60e	44	8048	A47	0.50	1	2	36	47	10103	...	0.427271	
4	54c1e3b9184cb5b6	44	9734	A47	0.00	1	2	36	47	10103	...	0.415349	

5 rows × 40 columns

In [127…]:

```
df
```

Out[127]:

	id	anchor	target	context	score	anchor_len	target_len	section	classes	context_desc	...	spacy_total
0	37d61fd2272659b1	44	46	A47	0.50	1	2	36	47	10103	...	0.000000
1	7b9652b17b68b7a4	44	257	A47	0.75	1	2	36	47	10103	...	0.388094
2	36d72442aefd8232	44	276	A47	0.25	1	2	36	47	10103	...	0.085102
3	5296b0c19e1ce60e	44	8048	A47	0.50	1	2	36	47	10103	...	0.427271
4	54c1e3b9184cb5b6	44	9734	A47	0.00	1	2	36	47	10103	...	0.415349
...	...	...	...	...	...	...	...	...	...	...	...	...
36468	8e1386cbefd7f245	26847	26860	B44	1.00	2	2	37	44	6191	...	1.779534
36469	42d9e032d1cd3242	26847	26861	B44	0.50	2	2	37	44	6191	...	1.673599
36470	208654ccb9e14fa3	26847	26865	B44	0.50	2	2	37	44	6191	...	1.483978
36471	756ec035e694722b	26847	26868	B44	0.75	2	2	37	44	6191	...	1.779534
36472	8d135da0b55b8c88	26847	26872	B44	0.50	2	2	37	44	6191	...	1.779534

36473 rows × 40 columns

In [128…]:

```
df.to_csv("final_data_file_with_encoding.csv",index=None)
```