Science Based Targets - illustrative data processing

Looking at the spreadsheet it would seem that the `Target` column contains a mix of info relating to near-term, long-term and net-zero targets. Assuming the idea is to split out the info into separate rows (i.e., for each company there should be a separate row for each of near-term, long-term and net-zero with the corresponding target text in that row), here is a rough example of how something might be done.

The basic approach is to separate out the sentences in the `Target` field and then search for target year values in each sentence. The output is then a table showing matches of sentences to target types. This can then be checked and corrected manually in Excel before producing the final table with target sentences married up to target type.

In [1]:
```
import numpy as np
import pandas as pd
import re
```

Load data

In [2]:
```
raw_data = pd.read_excel( 'data/companies-taking-action.xlsx', sheet_name = 'Worksheet'
)
```

In [3]:
```
raw_data
```

Out[3]:

	Company Name	ISIN	LEI	Near term - Target Status	Near term - Target Classification	Near term - Target Year	Long term - Target Status	C
0	(ACIP) Alexandria Company for Industrial Packages	NaN	NaN	Targets Set	Well-below 2°C	2030	NaN	
1	2degrees	NaN	72450039D2LOPG0Z2I07	Committed	NaN	NaN	NaN	
2	3B-Fibreglass	NaN	529900UNERQOV61CA912	Committed	NaN	NaN	NaN	
3	4most	NaN	NaN	Targets Set	Well-below 2°C	2030	NaN	
4	A&L Goodbody	NaN	NaN	Committed	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
4055	ZTO Express (Cayman) Inc.	US98980A1051	549300SCJPK3YZJTJR78	Committed	NaN	NaN	NaN	
4056	Zuellig Pharma	NaN	NaN	Committed	NaN	NaN	NaN	
4057	Zühlke Group	NaN	NaN	Committed	NaN	NaN	NaN	
4058	Zurich Insurance Group Ltd	CH0011075394	529900QVNRBND50TXP03	Committed	NaN	NaN	NaN	
4059	Ørsted	DK0060094928	W9NG6WMZIYEU8VEDOG48	Targets Set	1.5°C	2025	Targets Set	

4060 rows × 21 columns

Select unique identifier

Check that there are no duplicate companies. That way we know we can use company name as a unique identifier.

In [6]:
```
raw_data['Company Name'].duplicated().any()
```

Out[6]:
```
False
```

Create search terms from year data

Here we are taking of the year columns ('Near term - Target Year', 'Long term - Target Year', 'Net-Zero Year') and for each company extracting the years we want to search for in the `Target` column text.

In [7]:
```
target_year_columns = ['Near term - Target Year', 'Long term - Target Year', 'Net-Zero Year']
```

In [10]:
```
years =  pd.melt(raw_data, id_vars = ['Company Name'], value_vars = target_year_columns).sort_values(by = 'Company Name')
```

In [11]:
```
years
```

Out[11]:

	Company Name	variable	value
0	(ACIP) Alexandria Company for Industrial Packages	Near term - Target Year	2030
4060	(ACIP) Alexandria Company for Industrial Packages	Long term - Target Year	NaN
8120	(ACIP) Alexandria Company for Industrial Packages	Net-Zero Year	NaN
1	2degrees	Near term - Target Year	NaN
4061	2degrees	Long term - Target Year	NaN
...	...	...	...
8119	Ørsted	Long term - Target Year	2040
12179	Ørsted	Net-Zero Year	2040
4027	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	Near term - Target Year	2026
12147	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	Net-Zero Year	NaN
8087	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	Long term - Target Year	NaN

12180 rows × 3 columns

We should have a look at non-numeric values as these will need to be cleaned.

In [12]:
```
# Get unique non-NaN values.
unique_values = years['value'].dropna().unique()

# Select only non-numeric values.
non_numeric_values = filter(lambda value : not str(value).isnumeric(), unique_values)
print(list(non_numeric_values))
```

```
['FY2050', 'FY2030', '2030, 2024, 2030', '2030, 2034', 'FY2040', '2025, 2024', '2031,
2026', '2030, 2024', '2025, 2030', '2030, 2021', '2030, 2025', 'FY2031', 'FY2026',
'FY2045', 'FY2030/2031', '2032, 2025', '2030, 2035', '2024, 2025', '2028, 2025',
'FY2023, FY2030', '2025, 2023', '2022, 2030', '2030, 2022', '2030, 2020', 'FY2034',
'2027, 2025', '2025, 2022', '2035, 2025', '2022, 2027', '2026, 2030', '2034, 2024',
'FY2035', '2023, 2030', '2030, 2029', '2025, 2035', 'FY2025', '2030, 2023', '2030,
2026', 'F2030', '2030, 2020, 2025', 'FY2030/31', 'Y2026', '2025, 2029', '2023, 2024',
'2031, 2034', '2029, 2025', '2026, 2025', '2032, 2030, 2025', '2026, 2029', 'FY2028',
'2030, 2024, 2034', '2033, 2026', '2031, 2025', '2031, 2030', '2025, 2032', '2035/2030',
'2030, 2023, 2025', '2031, 2021, 2025']
```

It looks like we just have to handle the cases when:

- We have an FY (to indicate financial year presumably)
- We have a list of years separated by commas.
- We have years separated by a / (typically to indicate a financial year)

We can convert these into search terms.

In [13]:
```
def convert_year_field_to_search_terms(year_data):

    if pd.isnull(year_data):
        return np.NAN

    # Strip any spaces.
    year_data = str(year_data).replace(' ', '')

    # Split into separate terms when there is more than one.
    if ',' in year_data:
        terms = year_data.split(',')
    elif '/' in year_data:
        terms = year_data.split('/')
    else:
        terms = [year_data]

    # Add in the search term without the FY in case the FY is not included in the
Target text.
    for term in terms:
        if 'FY' in term:
            terms.append(term.replace('FY', ''))

    return terms
```

Let's test a couple of test cases.

In [14]:
```
print(convert_year_field_to_search_terms('FY2031, 2026'))
print(convert_year_field_to_search_terms('FY2030/31'))
```

```
['FY2031', '2026', '2031']
['FY2030', '31', '2030']
```

That seems to work OK. Let's now add in our search terms to the original data table (at the right-hand end).

In [15]:
```
for col in target_year_columns:
    raw_data[f"{col} search terms"] =
raw_data[col].apply(convert_year_field_to_search_terms)
```

In [16]:
```
raw_data
```

Near term - Target Status	Near term - Target Classification	Near term - Target Year	Long term - Target Status	Long term - Target Classification	Long term - Target Year	Net-Zero Committed	...	Location	Region	Sector
argets Set	Well-below 2°C	2030	NaN	NaN	NaN	No	...	Egypt	Africa	Containers and Packaging
ommitted	NaN	NaN	NaN	NaN	NaN	No	...	New Zealand	Oceania	Telecommunication Services
ommitted	NaN	NaN	NaN	NaN	NaN	Yes	...	Belgium	Europe	Consumer Durables Household and Personal Prod..
argets Set	Well-below 2°C	2030	NaN	NaN	NaN	No	...	United Kingdom (UK)	Europe	Professional Services
ommitted	NaN	NaN	NaN	NaN	NaN	No	...	Ireland	Europe	Professional Services
...	...	...	...	...	...	...	...	...	...	..
ommitted	NaN	NaN	NaN	NaN	NaN	No	...	China	Asia	Air Freight Transportation and Logistics
ommitted	NaN	NaN	NaN	NaN	NaN	No	...	Singapore	Asia	Healthcare Providers and Services, and Healthc..
ommitted	NaN	NaN	NaN	NaN	NaN	Yes	...	Switzerland	Europe	Software and Services
ommitted	NaN	NaN	NaN	NaN	NaN	No	...	Switzerland	Europe	Banks, Diverse Financials Insurance
argets Set	1.5°C	2025	Targets Set	1.5°C	2040	Yes	...	Denmark	Europe	Electric Utilities and Independent Power Produ..

Convert Target field to separate sentences

First we create a function to convert a paragraph to separate sentences.

In [17]:
```
def replace_periods(x):
    return x.group(0).replace('.', '|period|')

def extract_sentences(text):

    text = str(text)

    # Convert line breaks to periods
    text = text.replace('\n', '.')

    # Replace .org as this appears a lot.
    text = text.replace('.org', '|period|org')

    # Handle initials followed by periods as these are unlikely to be end of sentences.
    text = re.sub('[A-Z]\.', replace_periods, text)

    # Strip any spaces after periods.
    text = text.replace('. ', '.')

    # Split on periods
    sentences = text.split('.')

    # Put back periods for initials.
    sentences = list(map(lambda x : x.replace('|period|', '.'), sentences))

    # Remove empty string senctences
    return pd.Series(list(filter(lambda x : x != '', sentences)))
```

Test on an example sentence

In [18]:
```
test_sentence = raw_data['Target'][7]
print(test_sentence)
```

```
Overall Net-Zero Target
A.G. Barr plc commits to reach net-zero greenhouse gas emissions across the value chain by FY2050 from a FY2020 base year.
Near-Term Targets
A.G. Barr commits to reduce absolute scope 1 and 2 GHG emissions 60% by FY2030 from a FY2020 base year. A.G. Barr also commits to reduce absolute scope 3 GHG emissions from purchased goods and services, upstream transport and distribution and downstream transport and distribution 25% within the same timeframe.
Long-Term Targets
A.G. Barr plc commits to reduce absolute scope 1 and 2 GHG emissions 90% by FY2035 from a FY2020 base year. A.G. Barr plc also commits to reduce scope 3 GHG emissions from purchased goods and services, upstream transport and distribution and downstream transport  and distribution 90% by FY2050 from a FY2020 base year.
```

In [63]:
```
extract_sentences(test_sentence).to_list()
```

Out[63]:
```
['Overall Net-Zero Target',
 'A.G. Barr plc commits to reach net-zero greenhouse gas emissions across the value
chain by FY2050 from a FY2020 base year',
 'Near-Term Targets',
 'A.G. Barr commits to reduce absolute scope 1 and 2 GHG emissions 60% by FY2030 from a
FY2020 base year',
 'A.G. Barr also commits to reduce absolute scope 3 GHG emissions from purchased goods
and services, upstream transport and distribution and downstream transport and
distribution 25% within the same timeframe',
 'Long-Term Targets',
 'A.G. Barr plc commits to reduce absolute scope 1 and 2 GHG emissions 90% by FY2035
from  a FY2020 base year',
 'A.G. Barr plc also commits to reduce scope 3 GHG emissions from  purchased goods and
services, upstream transport and distribution and downstream transport  and distribution
90% by FY2050 from a FY2020 base year']
```

Create search table

Now we can create our search table which should contain for each company:

- columns for each of our target type search terms
- a column containing the separate sentences of the `Target` column

In [20]:
```
selected_cols = [
    'Company Name',
    'Near term - Target Year search terms',
    'Long term - Target Year search terms',
    'Net-Zero Year search terms',
    'Target'
]
```

In [43]:
```
sentence_table = raw_data[selected_cols].dropna(subset='Target')
```

In [44]:
```
sentence_table
```

Out[44]:

	Company Name	Near term - Target Year search terms	Long term - Target Year search terms	Net-Zero Year search terms	Target
0	(ACIP) Alexandria Company for Industrial Packages	[2030]	NaN	NaN	This target was approved using a streamlined t...
3	4most	[2030]	NaN	NaN	This target was approved using a streamlined t...
7	A.G. Barr plc	[FY2030, 2030]	[FY2050, 2050]	[FY2050, 2050]	Overall Net-Zero Target\nA.G. Barr plc commits...
9	A/S Vestfrost	[2030]	[2050]	[2050]	This target was approved using a streamlined t...
10	A1 Telekom Austria Group	[2030]	NaN	NaN	A1 Telekom Austria Group commits to reduce abs...
...	...	...	...	...	...
4047	Zhuhai Pilot Technology Co., Ltd.	[2030]	NaN	NaN	This target was approved using a streamlined t...
4049	Zimmer Biomet	[2030]	NaN	NaN	Zimmer Biomet commits to reduce absolute scope...
4050	Zimmermann	NaN	NaN	NaN	Zimmermann commits to reduce absolute scope 1 ...
4052	ZORDAN SRL SB	[2030]	[2050]	[2050]	This target was approved using a streamlined t...
4059	Ørsted	[2025]	[2040]	[2040]	Ørsted commits to reach net-zero greenhouse ga...

1954 rows × 5 columns

In [45]:
```
split_sentence_table = pd.concat(
    [sentence_table, sentence_table['Target'].apply(extract_sentences)],
    axis = 1
).drop(columns = 'Target')
```

In [47]:
```
id_cols = [
    'Company Name',
    'Near term - Target Year search terms',
    'Long term - Target Year search terms',
    'Net-Zero Year search terms'
]
search_table = (
    pd.melt(split_sentence_table, id_vars = id_cols)
    .sort_values(by = ['Company Name', 'variable'])
    .dropna(subset = 'value')
)
search_table
```

Out[47]:

	Company Name	Near term - Target Year search terms	Long term - Target Year search terms	Net-Zero Year search terms	variable	value
0	(ACIP) Alexandria Company for Industrial Packages	[2030]	NaN	NaN	0	This target was approved using a streamlined t...
1954	(ACIP) Alexandria Company for Industrial Packages	[2030]	NaN	NaN	1	https://sciencebasedtargets.org/faqs-for-smes/...
1	4most	[2030]	NaN	NaN	0	This target was approved using a streamlined t...
1955	4most	[2030]	NaN	NaN	1	https://sciencebasedtargets.org/faqs-for-smes/...
2	A.G. Barr plc	[FY2030, 2030]	[FY2050, 2050]	[FY2050, 2050]	0	Overall Net-Zero Target
...	...	...	...	...	...	...
29309	Ørsted	[2025]	[2040]	[2040]	14	*The target boundary includes land-related emi...
31263	Ørsted	[2025]	[2040]	[2040]	15	feedstocks
1938	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	0	Żabka commits to reduce absolute scope 1 and s...
3892	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	1	Żabka also commits to reduce scope 3 GHG emiss...
5846	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	2	Żabka commits that 75% of its suppliers by spe...

5902 rows × 6 columns

Now that we have a table with search terms and sentences in the right form, we can see where we have matches.

In [54]:
```
def match_search_terms(terms, sentence):
    if not isinstance(terms, list):
        return ''
    for term in terms:
        if term in sentence:
            return 'x'
    return ''
```

In [55]:
```
for col in target_year_columns:
    search_table[f"{col} match"] = search_table.apply(
```

```
        lambda x : match_search_terms(x[f"{col} search terms"], x['value']),
        axis = 1
    )
```

In [56]:
```
search_table
```

Out[56]:

	Company Name	Near term - Target Year search terms	Long term - Target Year search terms	Net-Zero Year search terms	variable	value	Near term - Target Year match	Lon term Targ Ye mat
0	(ACIP) Alexandria Company for Industrial Packages	[2030]	NaN	NaN	0	This target was approved using a streamlined t...		
1954	(ACIP) Alexandria Company for Industrial Packages	[2030]	NaN	NaN	1	https://sciencebasedtargets.org/faqs-for-smes/...	x	
1	4most	[2030]	NaN	NaN	0	This target was approved using a streamlined t...		
1955	4most	[2030]	NaN	NaN	1	https://sciencebasedtargets.org/faqs-for-smes/...	x	
2	A.G. Barr plc	[FY2030, 2030]	[FY2050, 2050]	[FY2050, 2050]	0	Overall Net-Zero Target		
...	...	...	...	...	...	...	...	
29309	Ørsted	[2025]	[2040]	[2040]	14	*The target boundary includes land-related emi...		
31263	Ørsted	[2025]	[2040]	[2040]	15	feedstocks		
1938	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	0	Żabka commits to reduce absolute scope 1 and s...	x	
3892	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	1	Żabka also commits to reduce scope 3 GHG emiss...		
5846	Żabka Polska Sp. z o. o. (Zabka Polska Sp. z o...	[2026]	NaN	NaN	2	Żabka commits that 75% of its suppliers by spe...	x	

5902 rows × 9 columns

We can now export this as csv. In Excel we can then run through by eye and tap in extra `x`s where needed. Then there'd be a bit more code to reconstruct the sentences correctly according to the matches.

In [60]:
```
search_table.to_csv('data/search-matches.csv', index = False)
```