Power Query (M): Remove English stop words from a text passage

Owen Price
flexyourdata.com, @flexyourdata

What are “stop words”?

Stop words are the words in a **stop list** (or ***stoplist*** or *negative dictionary*) which are filtered out (i.e. stopped) before or after processing of natural language data (text) because they are insignificant.[1] There is no single universal list of stop words used by all natural language processing tools, nor any agreed upon rules for identifying stop words, and indeed not all tools even use such a list. Therefore, any group of words can be chosen as the stop words for a given purpose. The "general trend in [information retrieval] systems over time has been from standard use of quite large stop lists (200–300 terms) to very small stop lists (7–12 terms) to no stop list whatsoever".[2]

https://en.wikipedia.org/wiki/Stop_word

1. *Rajaraman, A.; Ullman, J. D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 9781139058452*

2. *Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze (2008). Introduction to Information Retrieval. Cambridge University Press. p. 27.*

Python's NLTK provides a list of 179 English stop words

```python
from nltk.corpus import stopwords

print(f"There are {len(stopwords.words('english'))} English stopwords")
print("The first 10:")
print(stopwords.words('english')[:10])
```

[6] ✓ 0.0s

```
There are 179 English stopwords
The first 10:
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

Owen Price
flexyourdata.com, @flexyourdata

For this challenge, I saved the 179 stop words to a gist

GitHub Gist Search... All gists Back to GitHub

ncalm / nltk-english-stopwords.txt Secret Edit Delete Star 0

Created 2 days ago

Code Revisions 1 Download ZIP

List of 179 English stopwords from NLTK as of 2024-02-12

nltk-english-stopwords.txt Raw

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
```

Challenge: “Remove the stop words from this passage and return the list of words that appear more than once in the result”

	A	B	C	D	E	F	G	H	I
1	From this:								
2	Amidst the ancient oak's shelter, the forest floor cradled fallen leaves.								
3	Leaves of gold, crimson, and russet danced in the autumn breeze. Breeze								
4	whispered secrets to the trees, and the trees listened, their branches								
5	swaying in rhythm. Rhythm of life echoed through the woodland—a								
6									
7	symphony of rustling leaves and birdcalls. Birdcalls harmonized with the								
8	babbling brook, where water flowed over moss-covered stones. Stones,								
9									
10	worn smooth by time, held stories of ages past. Past and present								
11	intertwined, as sunbeams filtered through the canopy, dappling the earth.								
12									
13	Earth embraced memories, and memories lingered—a tapestry woven by								
14	seasons. Beneath the ancient oak, a squirrel gathered acorns, preparing for								
15									
16	winter.								

18	To this:	
19	**Word**	**Count**
20	leaves	3
21	stones	2
22	birdcalls	2
23	past	2
24	earth	2
25	memories	2
26	oak	2
27	ancient	2
28	breeze	2
29	rhythm	2
30	trees	2

Owen Price
flexyourdata.com, @flexyourdata

There were many great answers on the post. To read more, go to my profile and look at the recent posts

Or go here

Owen Price
flexyourdata.com, @flexyourdata

These slides describe the approach taken by Diarmuid Early

Owen Price
flexyourdata.com, @flexyourdata

Diarmuid Early (He/Him) • 1st 1d •••

Founder at Early Days Consulting, Partner at the Golden Company

Here's a formula approach - with the input text in a range named 'input' and the stop list in a range named 'stops'.

```
=LET(nonLetters,UNIQUE(TEXTSPLIT(input,,CHAR(SEQUENCE(26,,CODE("
A"))),1,1)),
  words,TEXTSPLIT(input,,nonLetters,1),
  nonStop,FILTER(words,ISERROR(XMATCH(words,stops))),
  counts,GROUPBY(nonStop,nonStop,ROWS,0,0,-2),
  output,FILTER(counts,INDEX(counts,0,2)>1),
  output)
```

For a fixed text like this, you could just go and look at all the punctuation marks to strip out manually, but if you want it to be flexible, generating a list of non-letters from the text to use to split on works better.

Then I split on all of those to get the list of words, matched against the list of stop words to remove those, used GROUPBY to get the unique words and counts, and filtered.

This seems easier than y
was easy until you told

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

A1 =SEQUENCE(26, , CODE("A"))

	A	B	C	D
1	65	=SEQUENCE(26, , CODE("A"))		
2	66			
3	67			
4	68			
5	69			
6	70			
7	71			
8	72			
9	73			
10	74			
11	75			
12	76			
13	77			
14	78			
15	79			
16	80			
17	81			
18	82			
19	83			
20	84			
21	85			
22	86			
23	87			
24	88			
25	89			
26	90			

To understand the formula, it's helpful to break it into pieces. The call to SEQUENCE starting at CODE("A") gives us a list of 26 character codes for the upper case alphabet

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1 =CHAR(SEQUENCE(26, , CODE("A")))

	A	B	C	D	E
1	A	=CHAR(SEQUENCE(26, , CODE("A")))			
2	B				
3	C				
4	D				
5	E				
6	F				
7	G				
8	H				
9	I				
10	J				
11	K				
12	L				
13	M				
14	N				
15	O				
16	P				
17	Q				
18	R				
19	S				
20	T				
21	U				
22	V				
23	W				
24	X				
25	Y				
26	Z				

Wrapping the sequence of character codes in the CHAR function gives us the upper case alphabet.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

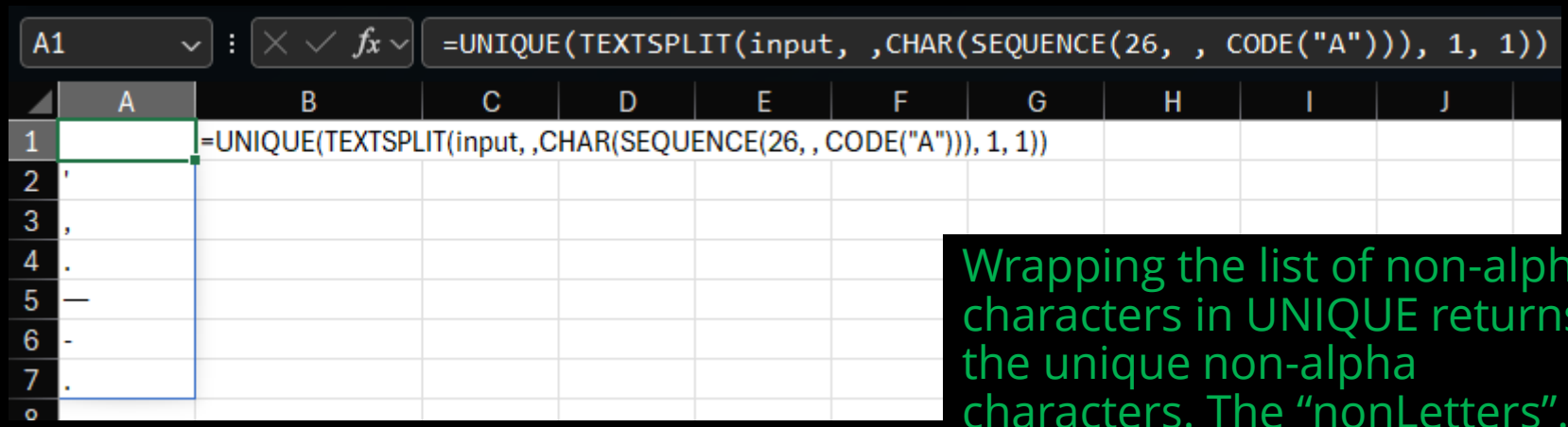
A1 =TEXTSPLIT(input, ,CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)

=TEXTSPLIT(input, ,CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)

This call to TEXTSPLIT uses the list of upper-case letters as the row-delimiter. The first “1” says “ignore empty results”. The second “1” says “use a case insensitive match”. The result is a list of all the non-alpha characters in the input text.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata



Wrapping the list of non-alpha characters in UNIQUE returns the unique non-alpha characters. The “nonLetters”.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

	A	B	C	D	E	F	G	H	I	J	K
1	Amidst	=LET(nonLetters, UNIQUE(TEXTSPLIT(input, ,CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)), TEXTSPLIT(input, , nonLetters, 1))									
2	the										
3	ancient										
4	oak										
5	s										
6	shelter										
7	the										
8	forest										
9	floor										
10	cradled										
11	fallen										
12	leaves										
13	Leaves										
14	of										
15	gold										

The “words” LET identifier is using “nonLetters” as the row-delimiter in another call to TEXTSPLIT to split the input text.

The result is a list of the tokens in the input text. These are tokens because this splits on apostrophes (see row 5, where the “s” is a separate token from “oak”).

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1

```
=LET(
  nonLetters, UNIQUE(TEXTSPLIT(input, ,CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
  words, TEXTSPLIT(input, , nonLetters, 1),
  HSTACK(words, XMATCH(words, stops)))
```

	A	B
1	Amidst	#N/A
2	the	63
3	ancient	#N/A
4	oak	#N/A
5	s	127
6	shelter	#N/A
7	the	63
8	forest	#N/A
9	floor	#N/A
10	cradled	#N/A

The call to XMATCH finds the position of stop words in the list of tokens. If a token is not a stop word, this returns the #N/A error.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1

```
=LET(
  nonLetters, UNIQUE(TEXTSPLIT(input, , CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
  words, TEXTSPLIT(input, , nonLetters, 1),
  FILTER(words, ISERROR(XMATCH(words, stops)))
)
```

	A
1	Amidst
2	ancient
3	oak
4	shelter
5	forest
6	floor
7	cradled
8	fallen
9	leaves
10	Leaves
11	gold
12	crimson

Wrapping the XMATCH in ISERROR returns a TRUE/FALSE array where TRUE means the word is not a stop word.

Passing this into FILTER returns a list of non-stop words.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1

```
=LET(
  nonLetters, UNIQUE(TEXTSPLIT(input, , CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
  words, TEXTSPLIT(input, , nonLetters, 1),
  nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
  GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2)
)
```

	A	B	C
1	leaves	3	
2	ancient	2	
3	birdcalls	2	
4	breeze	2	
5	earth	2	
6	memories	2	
7	oak	2	
8	past	2	
9	rhythm	2	
10	stones	2	
11	trees	2	
12	acorns	1	
13	ages	1	
14	Amidst	1	

Passing into GROUPBY the list of non-stop words as the first argument (row_fields) and the second argument (values) with ROWS as the function to apply to each group essentially creates a frequency table of the non-stop words. The 0, 0, -2 indicates “no field headers”, “no totals” and “sort in descending order”.

This is probably the most concise way to create a frequency table with one formula! 🥳

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1

```
=LET(
  nonLetters, UNIQUE(TEXTSPLIT(input, , CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
  words, TEXTSPLIT(input, , nonLetters, 1),
  nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
  counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
  HSTACK(INDEX(counts, 0, 2), INDEX(counts, 0, 2) > 1)
)
```

	A	B
1	3	TRUE
2	2	TRUE
3	2	TRUE
4	2	TRUE
5	2	TRUE
6	2	TRUE
7	2	TRUE
8	2	TRUE
9	2	TRUE
10	2	TRUE
11	2	TRUE
12	1	FALSE
13	1	FALSE
14	1	FALSE

INDEX(counts, 0, 2) indicates the second column of the frequency table (i.e. the counts themselves). Using this and 1 as operands to the “>” operator returns a TRUE/FALSE (Boolean) array which is TRUE if the count for a non-stop word is greater than 1.

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

A1

```
=LET(
  nonLetters, UNIQUE(TEXTSPLIT(input, , CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
  words, TEXTSPLIT(input, , nonLetters, 1),
  nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
  counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
  FILTER(counts, INDEX(counts, 0, 2) > 1)
)
```

	A	B
1	leaves	3
2	ancient	2
3	birdcalls	2
4	breeze	2
5	earth	2
6	memories	2
7	oak	2
8	past	2
9	rhythm	2
10	stones	2
11	trees	2

Using this Boolean array as the argument to FILTER removes the words which only appeared once!

```
=LET(
    nonLetters, UNIQUE(
        TEXTSPLIT(input, ,
            CHAR(SEQUENCE(26, , CODE("A"))), 1, 1)),
    words, TEXTSPLIT(input, , nonLetters, 1),
    nonStop, FILTER(words, ISERROR(XMATCH(words, stops))),
    counts, GROUPBY(nonStop, nonStop, ROWS, 0, 0, -2),
    FILTER(counts, INDEX(counts, 0, 2) > 1))
```

Owen Price
flexyourdata.com, @flexyourdata

Takeaways:

1. TEXTSPLIT is a powerful function – especially with arrays of delimiters
2. You can easily create a frequency table with GROUPBY
3. Follow Diarmuid and subscribe to @DimEarly on YouTube