

Plotting alignment data

```
In [1]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import json
import csv
import math
import pandas as pd
import utils.db_utils as db
import utils.plot_utils as plot
import utils.file_utils as file
import config

#####
# get configuration
cfg = config.getConfig() # configure values in config.js
#####

targetLang = cfg['targetLang']
bibleType = cfg['targetBibleType']
tWordsTypeList = cfg['tWordsTypeList']
dbPath = cfg['dbPath']
trainingDataPath = cfg['trainingDataPath']
testamentStr = cfg['testamentStr']
baseDataPath = cfg['baseDataPath']

type_ = 'kt'

In [2]: # get alignments for tW keyterms

minAlignments = 40
remove = ['ὁ', 'τὸ', 'τὰ']
alignmentsForWord, filteredAlignmentsForWord = db.fetchAlignmentDataForTWordCached(trainingDataPath, type_,
print(f"Original Language Alignments: {len(filteredAlignmentsForWord)}")

Using cached Alignments
Unfiltered Alignments: 4368
Filtered alignments by original list count is 105
Size of filtered alignments by original ./data/en/ult/TrainingData/kt_en_ult_NT_alignments_by_orig_40.json is
6.265 MB
Size of filtered alignments by original ./data/en/ult/TrainingData/kt_en_ult_NT_alignments_by_orig_40.csv is
1.679 MB
Filtered Alignments: 105
Original Language Alignments: 105
```

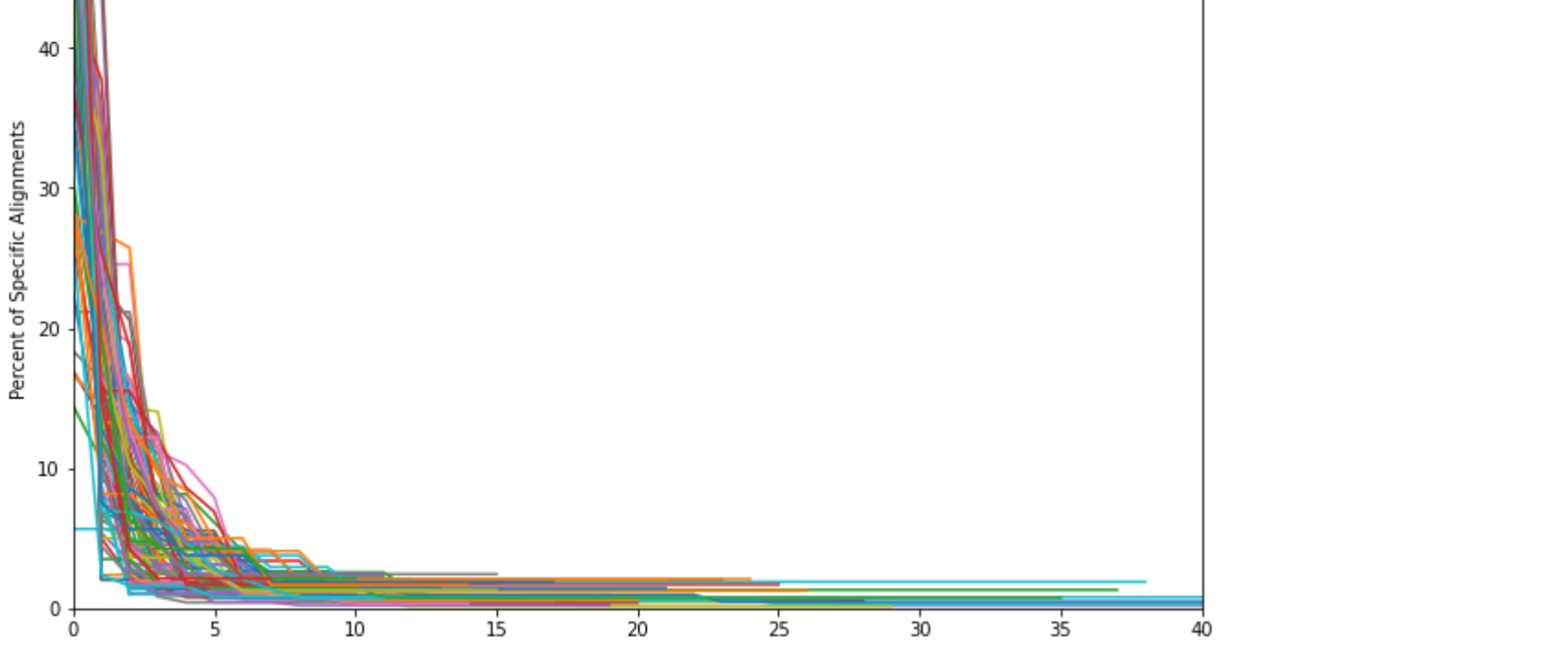
Analysis of alignments for tWord keyterms in the en_ult:

Frequency of alignments:

***Note that each line on the graphs below represents an alignment for a specific word. For example we have separate lines for 'Θεός'; 'Θεός'; or 'Θεοί' even though they have the same lemma. It made sense to group the alignments this way since aligners are likely to choose different target language words based on morphology of the word.

```
In [3]: frequenciesOfAlignments, stats = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, 'alignmentT

title = f"Plot of Variability of Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlim = [0, 40]
plotData = plot.plotFrequencies(frequenciesOfAlignments, title, ylabel, showXValues=False, xlim=xlim)
```



```
In [4]: stats = []
for word in frequenciesOfAlignments.keys():
    frequency = frequenciesOfAlignments[word]
    alignments = filteredAlignmentsForWord[word]
    total = len(alignments)
    counts = list(frequency.values)
    ratio = list(map(lambda x: x/total, counts))
    countsSeries = frequency.describe()
    alignments = filteredAlignmentsForWord[word]
    stats.append({
        'originalWord': word,
        'alignments': alignments,
        'frequency': frequency,
        'counts': counts,
        'countsSeries': countsSeries,
        'total': total,
        'ratio': ratio
    })

path_ = f'{trainingDataPath}/kt_en_ult_NT_alignments_stats.csv'
stats_ = pd.DataFrame(stats).drop(columns=['alignments'])
db.saveDataFrameToCSV(path_, stats_)
stats_.head()
```

	originalWord	frequency	counts	countsSeries	total	ratio
0	ἀμὴν	ἀμὴν = Truly 65 ἀμὴν = truly 33 ἀμ...	[65, 33, 1]	count 3.0 mean 33.0 std 32.0 min ...	99	[0.6565656565656566, 0.3333333333333333, 0.010...
1	ἄγγελος	ἄγγελος = angel 29 ἄγγελος = an angel...	[29, 15, 1, 1]	count 4.00000 mean 11.50000 std 1...	46	[0.6304347826086957, 0.32608695652173914, 0.02...
2	ἐποίησεν	ἐποίησεν = did 11 ἐποίη...	[11, 8, 8, 6, 5, 2, 2, 2, 2, 1, 1, 1, 1, ...]	count 38.00000 mean 2.0000000 std ...	76	[0.14473684210526316, 0.10526315789473684, 0.1...
3	ποιῆσαι	ποιῆσαι = to do 14 ποιῆσαι...	[14, 6, 3, 3, 3, 2, 1, 1, 1, 1, 1, 1, ...]	count 24.00000 mean 2.041667 std ...	49	[0.2857142857142857, 0.12244897959183673, 0.06...
4	ἐξουσίαν	ἐξουσίαν = authority ...	[31, 8, 6, 3, 2, 1, 1, 1, 1, 1, 1]	count 12.00000 mean 4.750000 std ...	57	[0.543859649122807, 0.14035087719298245, 0.105...

Analysis:

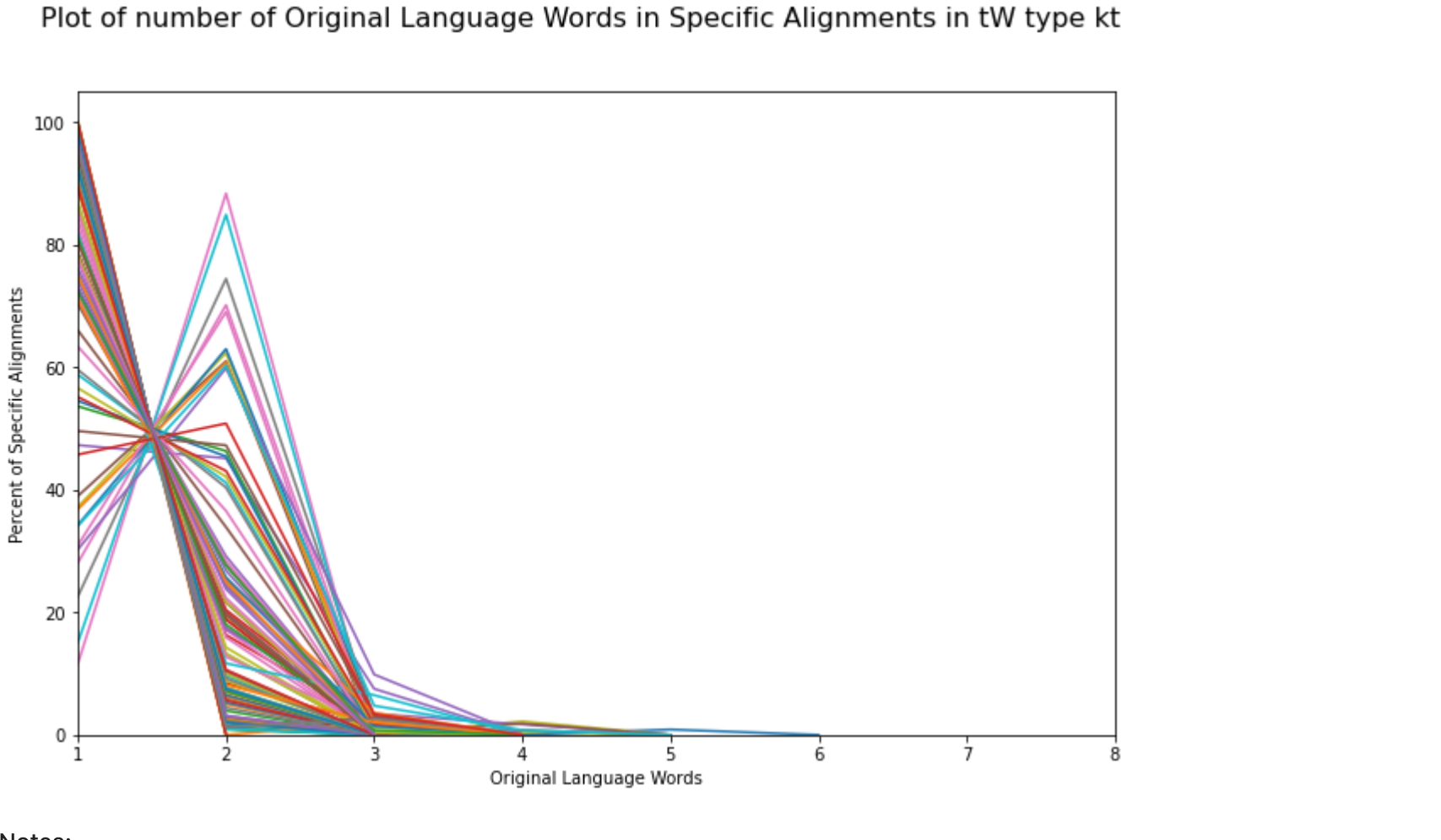
Analysis of numerical metrics:

Analysis of original language word count:

```
In [5]: field = 'origWordsCount'
field_frequencies, stats = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, field, sortIndex
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

print(f"Found {len(field_frequencies)} original language words for tW type {type_}")
title = f"Plot of number of Original Language Words in Specific Alignments in tW type {type_}"
ylabel = "Percent of Specific Alignments"
xlabel = "Original Language Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True, xlim=[1, 8])
```

Found 105 original language words for tW type kt
Plot of number of Original Language Words in Specific Alignments in tW type kt



Notes:

- this field analysis suggests that original word counts are tight - a threshold word count of 3 probably good for English to flag for review.

```
In [6]: threshold = 4
abnormalAlignments = {}
for origWord in field_frequencies:
    frequency = field_frequencies[origWord]
    count = len(frequency)
    if count >= threshold:
        abnormalAlignments[origWord] = frequency

print(f"Out of {len(field_frequencies)}, found {len(abnormalAlignments)} original language words that have i
filledFrequencies = db.zeroFillFrequencies(abnormalAlignments)

title = f"Plot of abnormal number of Original Language Words in Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Original Language Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True, xlim=[threshold, 10], ylim
```

Out of 105, found 3 original language words that have instances with over 4 words
Plot of abnormal number of Original Language Words in Specific Alignments in tW KeyTerms

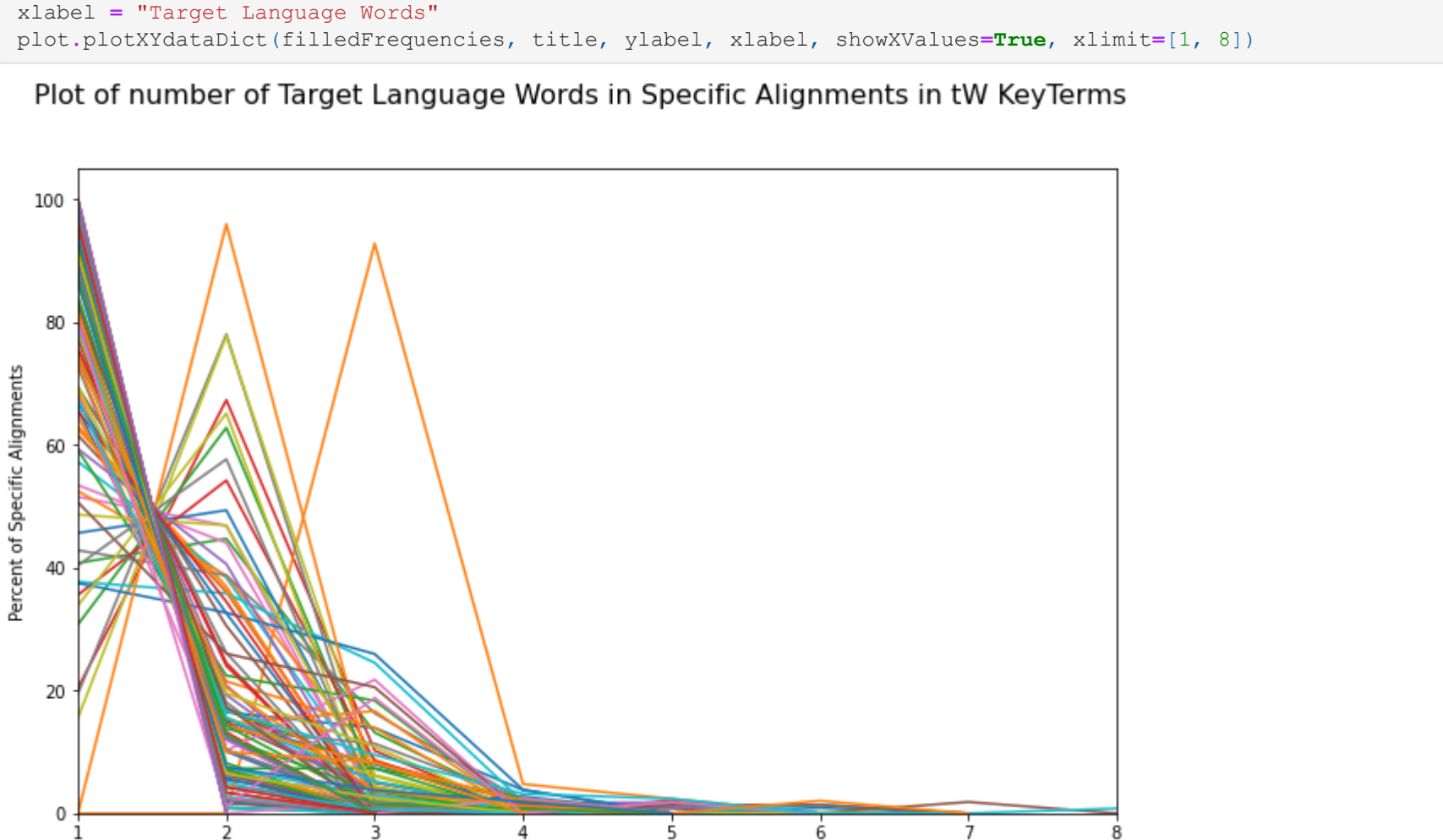


Analysis of target language word count:

```
In [7]: field = 'targetWordsCount'
field_frequencies, stats = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, field, sortIndex
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Target Language Words in Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Target Language Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True, xlim=[1, 8])
```

Plot of number of Target Language Words in Specific Alignments in tW KeyTerms



Notes:

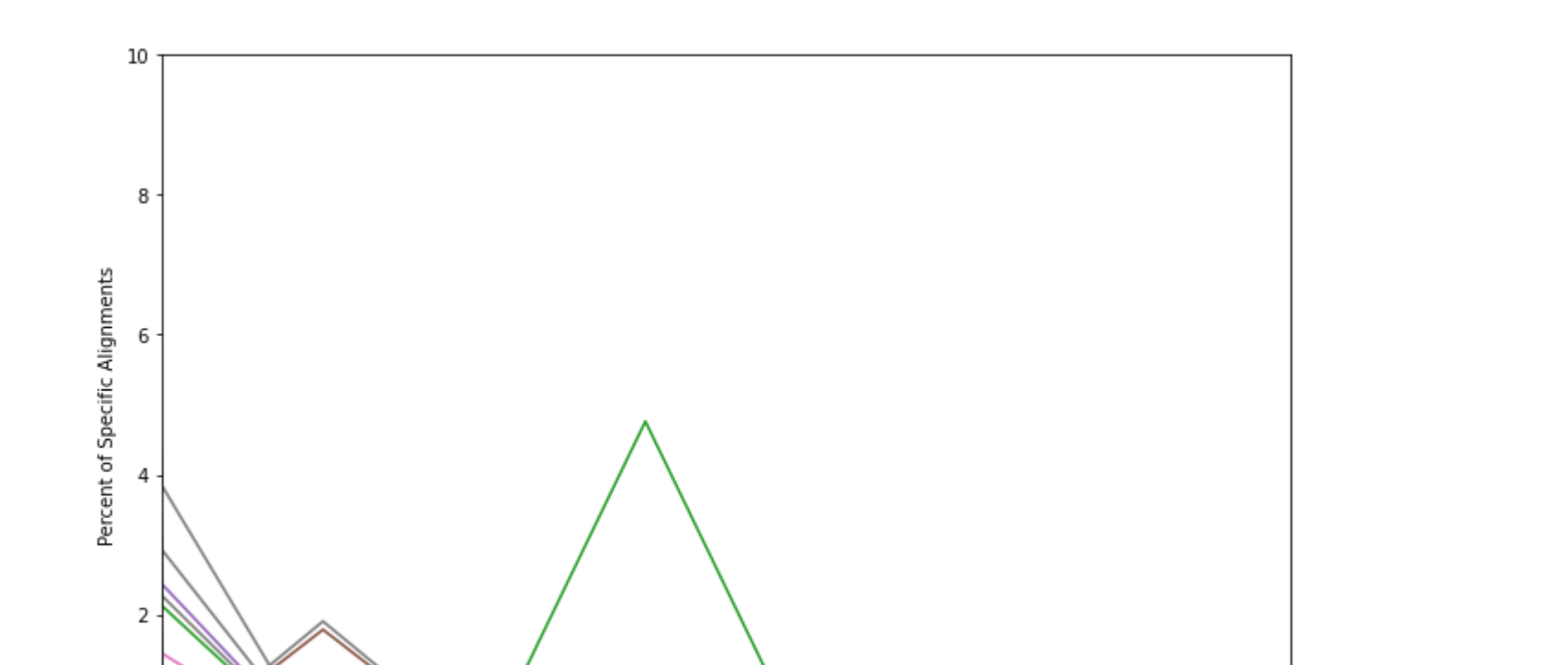
- this field analysis suggests that a threshold word count of 3 probably good for English to flag for review.

Analysis of count of extra unaligned words between aligned original language words:

```
In [8]: field = 'origWordsBetween'
field_frequencies, stats = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, field, sortIndex
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Extra Words in Discontiguous Original Language Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Extra Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True, xlim=[1, 8], ylim=[0,10])
```

Plot of number of Extra Words in Discontiguous Original Language Alignments in tW KeyTerms



Notes:

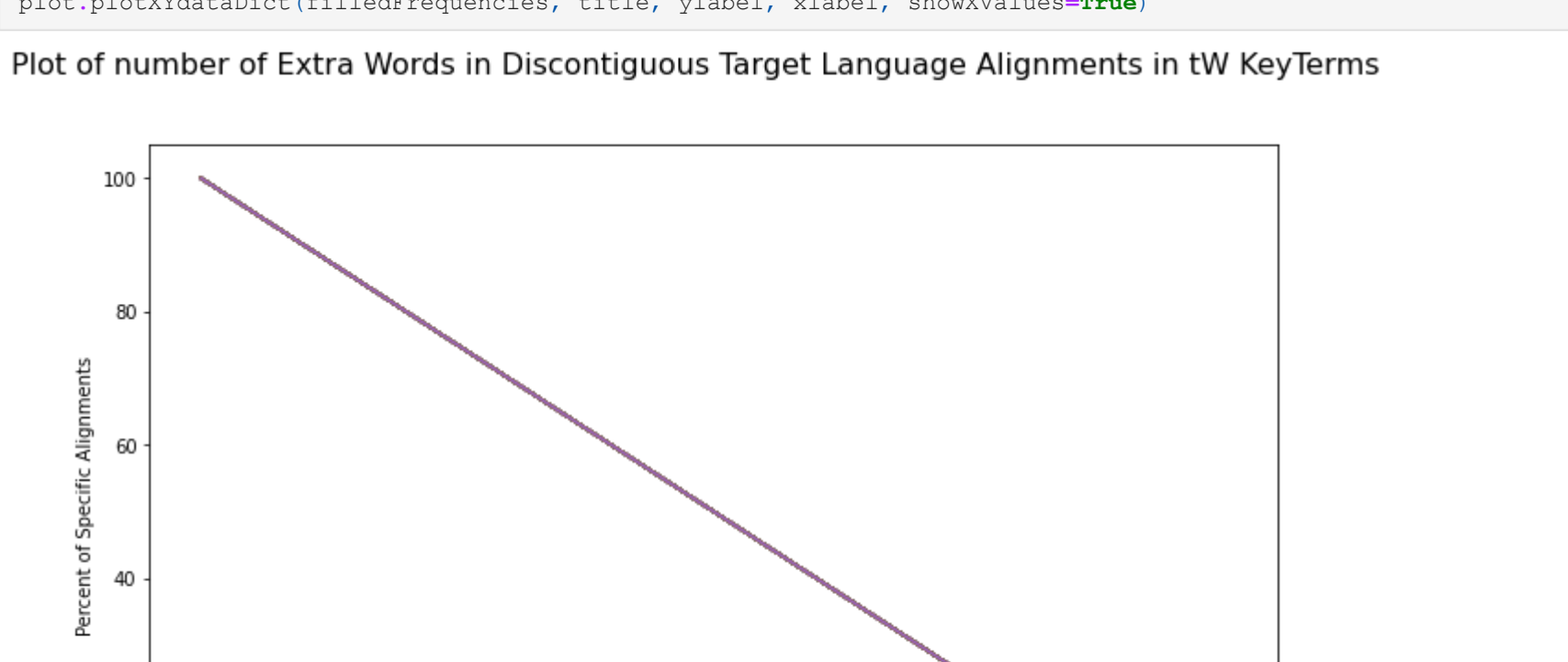
- this field analysis suggests that most original language alignments probably good. Probably the cases of a word between aligned words should be reviewed.

Analysis of count of extra unaligned words between aligned target language words:

```
In [9]: field = 'targetWordsBetween'
field_frequencies, stats = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, field, sortIndex
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Extra Words in Discontiguous Target Language Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Extra Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True)
```

Plot of number of Extra Words in Discontiguous Target Language Alignments in tW KeyTerms



Notes:

- this field analysis suggests that most target language alignments are very tight.

Generate CSV of questionable alignments:

```
In [10]: print(f"Testing tWords {type_} with minimum of {minAlignments} alignments")

alignmentOrigWordsThreshold = 3
alignmentTargetWordsThreshold = 5
origWordsBetweenThreshold = 1
targetWordsBetweenThreshold = 1
alignmentFrequencyMinThreshold = 5
warningPath = f'{baseDataPath}/{type_}_{bibleType}_{testamentStr}_warnings.json'
warningData = db.generateWarnings(warningPath, type_, bibleType, filteredAlignmentsForWord, alignmentOrigWor
alignmentTargetWordsThreshold, origWordsBetweenThreshold,
tag=f'(minAlignments)')
print(f"Found {len(warningData)} alignments to check - min threshold {minAlignments}")

Testing tWords kt with minimum of 40 alignments
Found 1151 alignments to check - min threshold 40
```

```
In [11]: print(f"Testing all tWords {type_}")
minAlignments = 0
alignmentsForWord, filteredAlignmentsForWord0 = db.fetchAlignmentDataForTWordCached(trainingDataPath, type_,
print(f"Original Language Alignments: {len(filteredAlignmentsForWord)}")

warningPath = f'{baseDataPath}/{type_}_{bibleType}_{testamentStr}_warnings.json'
warningData2 = db.generateWarnings(warningPath, type_, bibleType, filteredAlignmentsForWord0, alignmentOrigW
alignmentTargetWordsThreshold, origWordsBetweenThreshold,
tag=f'(minAlignments)')
print(f"Found {len(warningData2)} alignments to check - min threshold {minAlignments}")

Testing all tWords kt
Using cached Alignments
Unfiltered Alignments: 4368
Filtered alignments by original list count is 4344
Size of filtered alignments by original ./data/en/ult/TrainingData/kt_en_ult_NT_alignments_by_orig_0.json is
16.552 MB
Size of filtered alignments by original ./data/en/ult/TrainingData/kt_en_ult_NT_alignments_by_orig_0.csv is
4.478 MB
Filtered Alignments: 4344
Original Language Alignments: 105
Found 1630 alignments to check - min threshold 0
```

```
In [12]: basePath = f'./data/{type_}_{bibleType}_NT_summary'
summary = db.getStatsForAlignments(filteredAlignmentsForWord0)
df = pd.DataFrame(summary)
csvPath = basePath + '.csv'
summary_ = db.saveDictOfDictToCSV(csvPath, df)
print(f"saved summary of {len(summary_)} original words to {csvPath}")
summary_

saved summary of 4344 original words to ./data/kt_en_ult_NT_summary.csv
```

	id	originalWord	lemma	strong	alignmentsForOriginalWord	alignmentFrequency%	alignmentFreq	alignmentFr
0	ἀμὴν	ἀμὴν	ἀμὴν	G02810	99	{'ἀμὴν = Truly': 65.65656565656566, 'ἀμὴν = tr...	[65, 33, 1]	
1	ἄγγελος	ἄγγελος	ἄγγελος	G00320	46	{'ἄγγελος = angel': 63.04347826086957, 'ἄγγελο...	[29, 15, 1, 1]	
2	ἐποίησεν	ἐποίησεν	ποιέω	G41600	76	{'ἐποίησεν = did': 14.473684210526317, 'ἐποίησ...	[11, 8, 8, 6, 5, 2, 2, 2, 2, 1, 1, 1, 1, ...]	
3	ποιῆσαι	ποιῆσαι	ποιέω	G41600	49	{'ποιῆσαι = to do': 28.57142857142857, 'ποιῆσαι...	[14, 6, 3, 3, 3, 2, 1, 1, 1, 1, 1, 1, ...]	
4	ἐξουσίαν	ἐξουσίαν	ἐξουσία	G18490	57	{'ἐξουσίαν = authority': 54.385964912280706, '...	[31, 8, 6, 3, 2, 1, 1, 1, 1, 1, 1]	
...	
4339	λαός	λαός	λαός	G29920	5	{'λαός = people': 80.0, 'ὁ λαός = people': 20.0}	[4, 1]	
4340	λαοί	λαοί	λαός	G29920	1	{'λαοί = peoples': 100.0}	[1]	
4341	ἀντίχριστος	ἀντίχριστος	ἀντίχριστος	G05000	3	{'ἀντίχριστος = antichrist': 66.66666666666666, ...}	[2, 1]	
4342	ἀντίχριστοι	ἀντίχριστοι	ἀντίχριστος	G05000	1	{'ἀντίχριστοι = antichrists': 100.0}	[1]	
4343	ἀντίχρίστου	ἀντίχρίστου	ἀντίχριστος	G05000	2	{'ἀντίχρίστου = of': 50.0, 'ἀντίχρίστου = anti...	[1, 1]	

4344 rows x 42 columns

```
In [12]:
```

