Question 3.1 - How would you measure diversity?

First we look into the question, how we can measure the diversity of a subset of different answers. To calculate the diversity, we need to differentiate two types of answers: *categorical* and *continuous*.

An example for a categorical column is gender. We can measure the diversity in this case with the *entropy* measure. Since entropy measures how spread out or balanced the selected values are, it is ideal. In the end we want to normalize the entropy to account for classes that are not in the selected subset.

For continuous data columns like *age* we can measure the diversity using the *standard deviation*. We could also use the *entropy* and treat every age as a separate class. However, this would mean that we don't account for the absolute age difference between workers, just for different ages. We therefore decided that the standard deviation is the better metric. To get a score in-between 0 and 1, it has to be normalized: The std. deviation of the subset is divided by the std. deviation of the maximum possible std. deviation of the full dataset. This is done by calculating the difference of the maximum and minimum values of the original dataset and then dividing it by two.

In the end we calculate our total diversity score by calculating the diversity of each relevant column and then generating their mean.

Excerpt of the original dataframe:

Out[3]:

	Answer.ID	FailingMethod	Question.ID	Answer.duration	Answer.confidence
0	261	HIT01_8	0	90.984	4
1	262	HIT01_8	0	133.711	5
2	263	HIT01_8	0	77.696	5
3	264	HIT01_8	0	46.644	1
4	265	HIT01_8	0	215.416	5
...	...	...	...	...	...
2575	2316	HIT08_54	128	220.420	2
2576	2317	HIT08_54	128	322.790	4
2577	2318	HIT08_54	128	159.530	5
2578	2319	HIT08_54	128	68.578	5
2579	2320	HIT08_54	128	72.605	4

2580 rows × 26 columns

For our analysis we pick the Failing Method "HIT01_8" and look at question no. 1. We then filter out incorrect answers and are left with 12 correct answers, given to question 1.

Out[4]:

	Answer.ID	FailingMethod	Question.ID	Answer.duration	Answer.confidence	A
20	441	HIT01_8	1	140.407	4	
22	443	HIT01_8	1	112.409	5	
23	444	HIT01_8	1	76.418	5	
24	445	HIT01_8	1	261.928	5	
25	446	HIT01_8	1	236.045	5	
27	448	HIT01_8	1	259.790	3	
28	449	HIT01_8	1	230.142	4	
29	450	HIT01_8	1	253.302	4	
30	451	HIT01_8	1	89.521	4	
34	455	HIT01_8	1	161.419	4	
35	456	HIT01_8	1	58.673	5	
36	457	HIT01_8	1	129.491	5	

12 rows × 26 columns

For demonstration we can print the gender diversity and its normalized value of all picked answers compared to the full dataset:

```
def categorical_diversity(series):
    counts = series.value_counts(normalize=True)
    return entropy(counts, base=2)


def categorical_diversity_norm(series, series_full):
    k = series_full.nunique()
    return categorical_diversity(series) / (np.log2(k) if k > 1
else 1)
```

```
Gender diversity of all answers: 0.8112781244591328
Normalized gender diversity 0.5118595071429148
```

For continuous variables, we use the following method:

```
def coefficient_variation(series, series_full):
    if len(series) < 2:
        return 0
    return series.std() / ((series_full.max() -
series_full.min()) / 2)
```

For demonstration purposes we print the age diversity of all picked answers compared to the full dataset:

```
Normalized age diversity 0.3035038076373404
```

For our analysis, we chose the following categorical columns

- Worker.profession
- Worker.gender

and the following continuous columns

- Worker.age
- Worker.yearsOfExperience

To combine multiple factors into a single diversity score, we take both categorical and continous columns, give each an equal weight and sum them up. The diversity score is then normalized by the number of factors.

The combined diversity score of our picked answers compared to the full dataset:

```
Combined Diversity 0.4759271292774812
```

Subsets

We are now going to look at subsets of the 12 previously picked answers. We will evaluate these subset according to three different measures:

- diversity (regarding in regard to persons)
- readability
- semantic similarity of the subset to our hand-crafted ground truth

To check for similarity, we embed the two sentences to be compared with the model "all-MiniLM-L6-v2" and use cosine similarity on the transformed embeddings.

```python
def calculate_cosine_similarity(hyp, ref):
    hyp_embedding = model.encode(hyp)
    ref_embedding = model.encode(ref)
    return np.dot(hyp_embedding, ref_embedding) /
(np.linalg.norm(hyp_embedding) * np.linalg.norm(ref_embedding))
```

To calculate a readability score, we are using the Flesch Reading Ease formula.

Since the full dataset of answers is relatively small (12), we can search through all possible subsets of lengths 1 to 12 and compare their scores in all three categories. Below are the number of combinations to check for a subset of size `n`

Below is an excerpt of the three scores for each possible subset. The text-content to be scored per subset is defined by the concatenation of the explanations of the subset. Keep in mind, that no permutations are included. The order is random.

Out[17]:

	readability	similarity	diversity
0	80.62	0.722904	0.000000
1	69.45	0.784441	0.000000
2	48.47	0.801706	0.000000
3	39.71	0.786183	0.000000
4	28.84	0.870643	0.000000
...	...	...	...
4090	59.33	0.837821	0.681664
4091	60.14	0.818906	0.734276
4092	59.94	0.814052	0.737129
4093	59.43	0.794701	0.655354
4094	58.72	0.809571	0.718752

4095 rows × 3 columns

Pareto Fronts

To understand which scores can be maximized while taking loss of other scores into account, we can use pareto fronts.

At first, we can remove any dominated solutions from the dataframe. A solution is dominated if there is another solution that is better in all scores.

Out[23]:

	readability	similarity	diversity	readability.norm	similarity.norm	diversity.n
840	66.27	0.817777	0.883779	0.848372	0.879052	0.900
21	77.87	0.697944	0.981933	0.951291	0.666715	1.000
1107	71.75	0.794625	0.863310	0.896992	0.838028	0.879
442	72.87	0.791136	0.847902	0.906929	0.831846	0.863
362	66.78	0.797721	0.887771	0.852897	0.843514	0.904
...	...	...	...	...	...	
556	72.97	0.790803	0.501181	0.907817	0.831257	0.510
293	75.81	0.736891	0.514847	0.933014	0.735727	0.524
192	48.64	0.886034	0.258099	0.691953	1.000000	0.262
0	80.62	0.722904	0.000000	0.975690	0.710942	0.000
10	83.36	0.596953	0.000000	1.000000	0.487764	0.000

83 rows × 7 columns

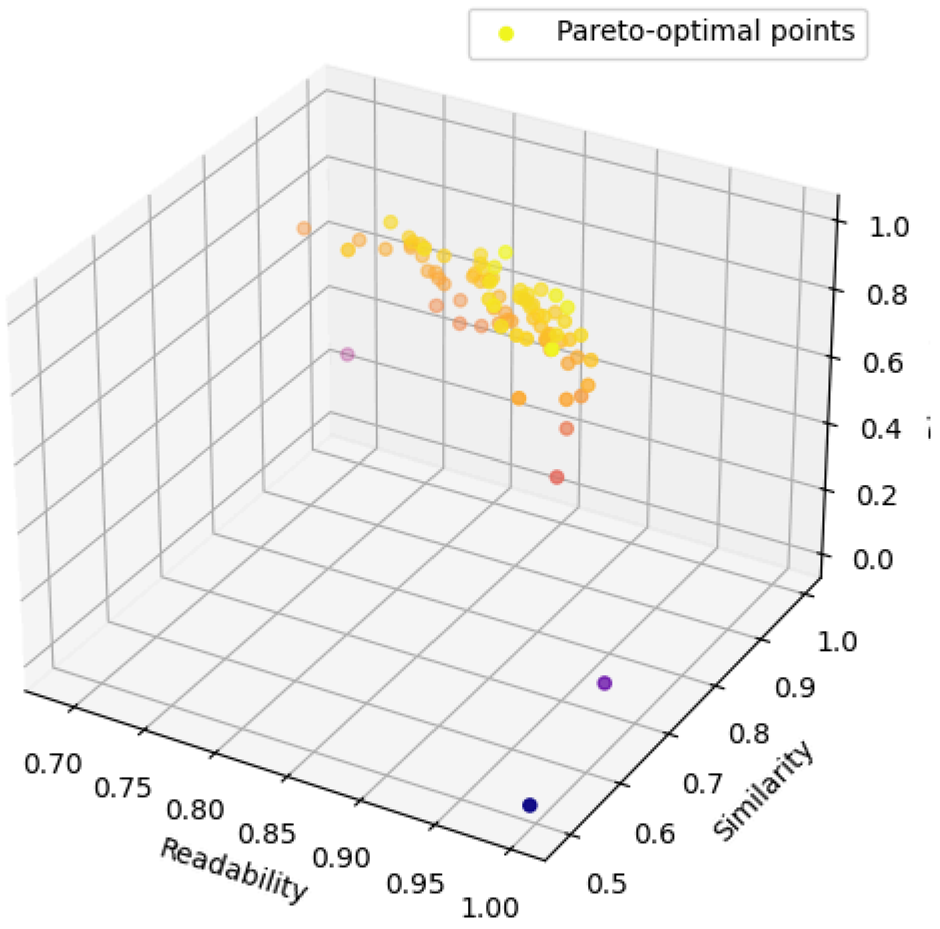
We reduced the number of interesting subsets from 4095 to 83.

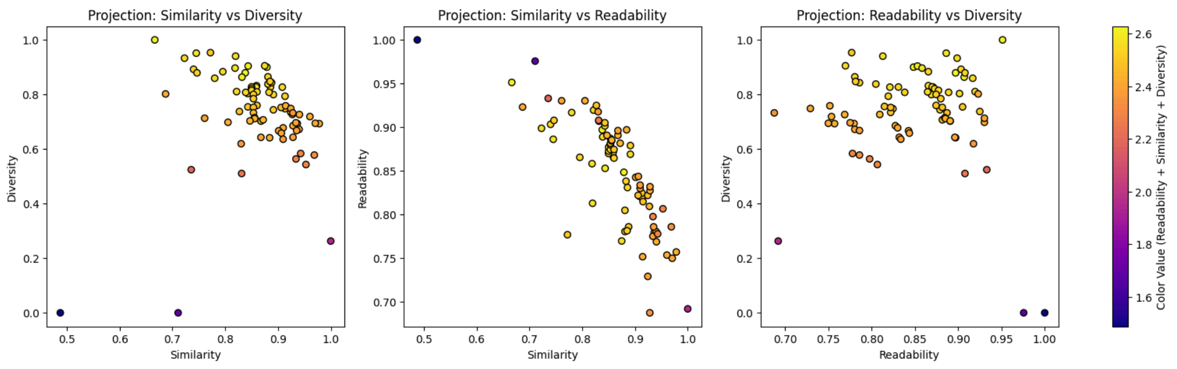
Ordered by the total score we see that the best score is 0.876. Printed is the best possible concatenated explanation, regarding all three scores:

```
Minutes are set to -15; which is less then 0 and it throws illegal arg exc
eption

In the code there is a check that 0 <= minutes < 60 and the minutesOffset
is -15 which does not fall into these prarmeters thus throwing an Exceptio
n

YES. The issue is on line 279 (as I explained in my first question; of whi
ch I misunderstood that I was only being asked about the specific issue; n
ot generalized issue). On line 279 the variable "minutesOffSet" is paramet
erized to throw an exception if it is < 0 or > 59. Line 279 should read "i
f (minutesOffset < -59 || minutesOffset > 59) {" because now the method ca
n take in the number of minutes as a negative and will allow the method to
properly progress to invoke/call further methods such as those asked about
in the two previous questions.

This variable contains a value of -15 as set by DateTimeZone.forOffsetHour
sMinutes(-2; -15). Line 279 checks to see if is a valid value; meaning tha
t is between 0 and 59. Since it is not; an exception error is thrown in li
ne 280.

yep; they are checking if minutesOffset < 0 to throw an exception; and as
-15 <0; it gets thrown. looks like they updated the comments but not the c
ode. and this is why comments are evil liars that can't be trusted!
```

For reference, here ist the chosen ground truth from Task 2:

> "The IF statement in line 279 checks whether minutesOffset is set to a value between 0 and 59. If not, an IllegalArgumentException is thrown. This is a bug because the minutesOffset may also be negative. The IF statement should check for the minutesOffset to be between -59 and 59.""





3.2

Question:
What is the max readability and semantic similarity independent of the diversity?

We use the average of the two norms as score.

`df_results['score_similarity_readability'] = (df_results['readability.norm'] + df_results['similarity.norm']) / 2`

Out[29]:

	readability	similarity	diversity	readability.norm	similarity.norm	diversity.r
117	71.75	0.820717	0.629936	0.896992	0.884262	0.64
2615	69.72	0.824718	0.730321	0.878981	0.891352	0.74
2409	71.65	0.811413	0.631291	0.896105	0.867775	0.64

Indeed, by sacrificing diversity we can achieve a higher score fo readability and similarity: 0.891

3.3

Question
What is the max diversity for (previously achieved) max semantic similarity (compromising readability)?

We sort the list by the similarity norm:

Out[30]:

	readability	similarity	diversity	readability.norm	similarity.norm	diversity.no
192	48.64	0.886034	0.258099	0.691953	1.000000	0.262
34	30.03	0.882031	0.257583	0.526839	0.992908	0.262
315	55.98	0.873746	0.680608	0.757076	0.978227	0.693
186	49.96	0.873167	0.572442	0.703664	0.977201	0.582
4	28.84	0.870643	0.000000	0.516281	0.972729	0.000

Sorting by normed similarity and looking at the perfect maximum of 1.000, we get a diversity of 0.258.

Compromising slightly on similarity, picking the third entry at 0.978, we can achieve a diversity score of 0.693.