A COMMONALITIES IN ANNOTATIONS

This appendix highlights the most insightful overlaps between annotations of samples (predictions). In this context, insightful is defined as being an unexpectedly high percentage of overlap between two (sub)categories of annotations. Percentages below 5% are immediately disregarded for the results of the commonalities. The comparisons made are:

- CodeGPT vs. UniXcoder - see Table 8 (relation of singular annotations between models in CodeGPT's perspective)
- UniXcoder vs. CodeGPT - see Table 9 (relation of singular annotations between models in UniXcoder's perspective)
- CodeGPT vs. CodeGPT - see Table 10 (relation of samples with multiple annotations for CodeGPT)
- UniXcoder vs UniXcoder - see Table 11 (relation of samples with multiple annotations for UniXcoder)

The results of these comparisons form the argument for the results mentioned in section 6.2 and 7.2. For the complete results, please refer to the repository linked to this project, which contains the raw data (see `.json`-files in `/models/evaluation/annotated/output`) and scripts to manipulate the data, allowing to get this output.

Table 8: Insightful overlaps between annotations of samples between CodeGPT and UniXcoder.

CodeGPT annotation	UniXcoder annotation	Overlap
undefined	undefined	100.00% (1/1)
empty	incomplete	54.72% (58/106)
incomplete	incomplete	43.33% (13/30)
empty	wrong syntax	25.47% (27/106)
wrong syntax	wrong syntax	23.08% (3/13)
incomplete	wrong syntax	20.00% (6/30)
extra comment	incomplete	16.81% (19/113)
wrong syntax	incomplete	15.38% (2/13)
valid	wrong syntax	15.00% (3/20)
empty	undefined	14.15% (15/106)
valid	valid	10.00% (2/20)
valid	undefined	10.00% (2/20)

Table 9: Insightful overlaps between annotations of samples between UniXcoder and CodeGPT.

UniXcoder annotation	CodeGPT annotation	Overlap
empty	empty	100.00% (1/1)
variable definition	variable definition	73.68% (14/19)
undefined	empty	48.39% (15/31)
incomplete	empty	44.62% (58/130)
valid	extra comment	33.33% (6/18)
wrong syntax	empty	28.12% (27/96)
valid	valid	11.11% (2/18)
incomplete	incomplete	10.00% (13/130)
wrong syntax	incomplete	6.25% (6/96)

Table 10: Insightful overlaps between annotations of samples between CodeGPT itself. The base filter was a minimal overlap of at least 10% and 2 overlaps. Furthermore, the same (sub)category was also disregarded immediately or when "`extra comment`" was annotated, as the latter requires a standalone analysis.

CodeGPT annotation	CodeGPT annotation	Overlap
empty	case expressions, body	77.36% (82/106)
functions, complete	wrong function	75.00% (9/12)
lists, range	generators, body	51.52% (17/33)
generators, body	lists, range	39.53% (17/43)
case expressions, body	empty	36.94% (82/222)
functions, body	wrong function	32.14% (9/28)
variable definition	wrong value	28.57% (8/28)
guards = |, body	wrong value	27.50% (11/40)
lists, ++	wrong function	27.27% (3/11)
logical operators, &&	wrong value	27.27% (3/11)
wrong value	case expressions, body	27.14% (19/70)
lists, list comprehension	wrong function	26.83% (11/41)
logical operators, ==	wrong function	26.67% (4/15)
wrong function	case expressions, body	26.36% (29/110)
wrong type	case expressions, body	25.00% (3/12)
wrong syntax	lists, list comprehension	23.08% (3/13)
if/then/else, else	wrong function	21.43% (6/28)
logical operators, ==	wrong value	20.00% (3/15)
incomplete	functions, arguments	20.00% (6/30)
incomplete	case expressions, body	20.00% (6/30)
functions, arguments	wrong function	19.59% (19/97)
lists, list comprehension	generators, body	19.51% (8/41)
generators, body	lists, list comprehension	18.60% (8/43)
lists, range	lists, list comprehension	18.18% (6/33)
wrong function	functions, arguments	17.27% (19/110)
wrong value	arithmetic logic	17.14% (12/70)
incomplete	generators, body	16.67% (5/30)
wrong value	guards = |, body	15.71% (11/70)
if/then/else, then	wrong value	14.81% (4/27)
variable definition	wrong function	14.29% (4/28)
variable definition	empty	14.29% (4/28)
arithmetic logic	wrong value	14.29% (12/84)
case expressions, body	wrong function	13.06% (29/222)
wrong value	functions, arguments	12.86% (9/70)
guards = |, body	empty	12.50% (5/40)

Table 11: Insightful overlaps between annotations of samples between UniXcoder itself. The base filter was a minimal overlap of at least 10% and 2 overlaps. Furthermore, the same (sub)category was also disregarded immediately.

UniXcoder annotation	UniXcoder annotation	Overlap
arithmetic operators, mod	wrong value	75.00% (3/4)
undefined	case expressions, body	67.74% (21/31)
functions, complete	wrong function	66.67% (4/6)
logical operators, ||	incomplete	66.67% (4/6)
if/then/else, all	incomplete	62.50% (5/8)
incomplete	case expressions, body	58.46% (76/130)
variable name	case expressions, body	57.14% (8/14)
lists, ++	wrong value	50.00% (6/12)
wrong syntax	case expressions, body	44.79% (43/96)
wrong syntax	incomplete	39.58% (38/96)
wrong type	case expressions, body	38.46% (10/26)
wrong function	case expressions, body	38.27% (31/81)
lists, range	generators, body	37.93% (11/29)
lists, range	wrong value	37.93% (11/29)
arithmetic logic	case expressions, body	37.93% (33/87)
if/then/else, all	wrong syntax	37.50% (3/8)
case expressions, body	incomplete	31.80% (76/239)
variable definition	incomplete	31.58% (6/19)
variable definition	arithmetic logic	31.58% (6/19)
generators, body	lists, list comprehension	30.23% (13/43)
incomplete	wrong syntax	29.23% (38/130)
lists, list comprehension	generators, body	27.66% (13/47)
generators, body	lists, range	25.58% (11/43)
logical operators, &&	incomplete	25.00% (3/12)
wrong function	wrong syntax	24.69% (20/81)
wrong type	functions, arguments	23.08% (6/26)
wrong value	case expressions, body	22.73% (15/66)
wrong value	arithmetic logic	22.73% (15/66)
functions, body	wrong function	22.58% (7/31)
functions, body	arithmetic logic	22.58% (7/31)
guards = |, body	wrong syntax	22.50% (9/40)
valid	generators, body	22.22% (4/18)
wrong syntax	wrong function	20.83% (20/96)
lists, range	incomplete	20.69% (6/29)
guards = |, body	wrong value	20.00% (8/40)
lists, list comprehension	case expressions, body	19.15% (9/47)
lists, list comprehension	incomplete	19.15% (9/47)
functions, arguments	wrong syntax	18.18% (16/88)
case expressions, body	wrong syntax	17.99% (43/239)
guards = |, body	incomplete	17.50% (7/40)
guards = |, body	arithmetic logic	17.50% (7/40)
wrong function	functions, arguments	17.28% (14/81)
arithmetic logic	wrong value	17.24% (15/87)
wrong value	lists, range	16.67% (11/66)
valid	lists, list comprehension	16.67% (3/18)
valid	case expressions, body	16.67% (3/18)
wrong syntax	functions, arguments	16.67% (16/96)
generators, body	wrong value	16.28% (7/43)
case expressions, body	arithmetic logic	13.81% (33/239)
wrong function	incomplete	13.58% (11/81)
case expressions, body	wrong function	12.97% (31/239)
lists, list comprehension	wrong syntax	12.77% (6/47)
wrong value	guards = |, body	12.12% (8/66)
arithmetic logic	incomplete	11.49% (10/87)
wrong value	generators, body	10.61% (7/66)

B CODEGPT'S EXTRA COMMENTS

Upon further analysis of the results in other appendices, it was observed that the "`extra comment`" annotation in CodeGPT predictions was used substantially – in fact, UniXcoder has predicted a comment 0 times. Furthermore, the comment did not add any semantic value and seemed extremely similar to other predictions. Hence, the following analyses have been done:

- The distribution of the notable (sub)categories of the samples that were annotated with "`extra comment`" to check whether there is a pattern (see Figure 4) – no clear pattern has been revealed, however, it does show that there are quite some predictions that are not considered an *Exact Match* (EM) due to the added comment, yet are still *valid*, as annotated.
- The uniqueness of the "`extra comment`" by counting duplicates (see Table 12) – there is definitely a pattern: all extra comments follow the same sentence structure with only a different value mentioned, which is also highly similar to the other values.

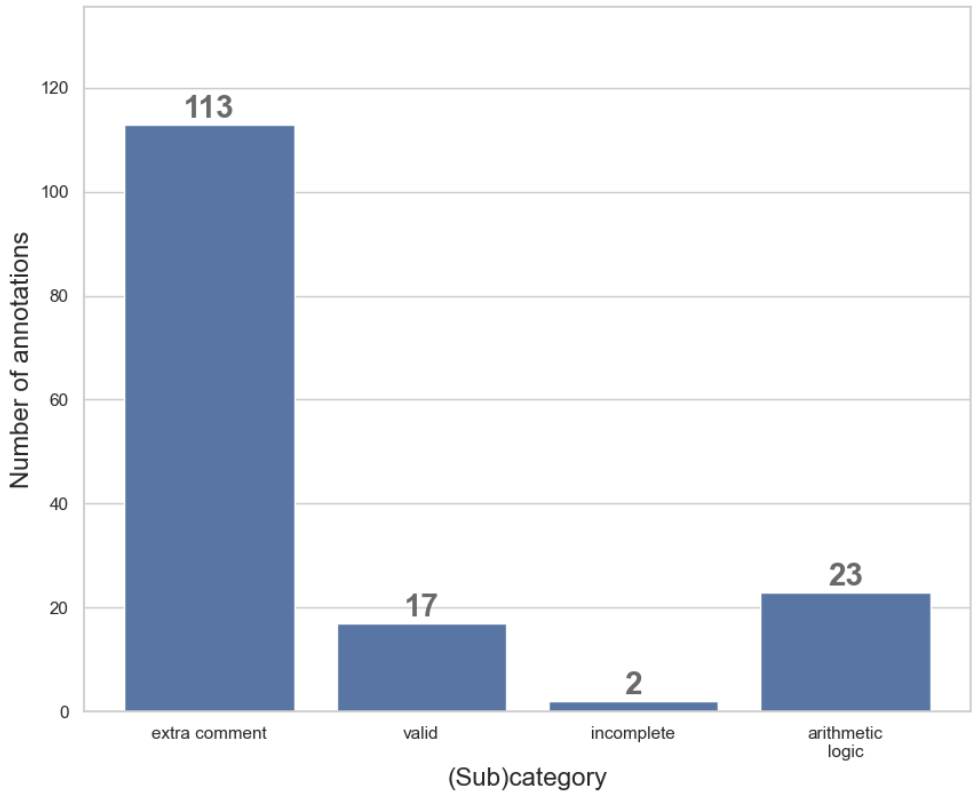

Figure 4: Distribution of notable CodeGPT's annotation (sub)categories for samples marked with "`extra comment`".

Table 12: Count of unique comments of CodeGPT's predictions based on the 113 "`extra comment`" annotations.

Comment	Counter
"| Creates a value of 'ProjectsLocationsDataSetsConsentStoresConsentsRevoke' with the minimum fields required to make a request."	62
"| Creates a value of 'ProjectsLocationsDataSetsFhirStoresFhirUpdate' with the minimum fields required to make a request."	27
"| Creates a value of 'ProjectsLocationsDataSetsFhirStoresFhirConditionalUpdate' with the minimum fields required to make a request."	15
""	5
"| Creates a value of 'ProjectsLocationsDataSetsFhirStoresFhirConditionalPatch' with the minimum fields required to make a request."	3
"| Creates a value of 'ProjectsLocationsDataSetsHl7V2StoresMessagesList' with the minimum fields required to make a request."	2
"| Creates a value of 'ProjectsLocationsAgentsEnvironmentsExperimentsPatch' with the minimum fields required to make a request."	2
"| Creates a value of 'ProjectsLocationsDataSetsConsentStoresConsentsList' with the minimum fields required to make a request."	1
"| Creates a value of 'ProjectsLocationsDataSetsFhirStoresFhirRead' with the minimum fields required to make a request."	1
"| Creates a value of 'GoogleCloudVideointelligenceV1p1beta1_ObjectTrackingAnnotation' with the minimum fields required to make a request."	1
"| Creates a value of 'ProjectsLocationsDataSetsDicomStoresStudiesSeriesRetrieveStudy' with the minimum fields required to make a request."	1

C DISTRIBUTION OF ANNOTATIONS

In this appendix, an overview is given of the distribution of the manual annotations made. There will be several comparisons, see Table 13 below, to reach as many insightful conclusions for the common pitfalls of the language models, as described in section 6.2 and 7.2.

Table 13: Figure overview of the comparisons made between the specific distributions of annotations.

Model	Filter	Model	Filter	Figure
CodeGPT	¬EM ∪ ¬valid	UniXcoder	¬EM ∪ ¬valid	Figure 5
CodeGPT	EM ∩ valid	UniXcoder	EM ∩ valid	Figure 6
CodeGPT	¬EM ∪ ¬valid	CodeGPT	– (no filter)	Figure 7
CodeGPT	EM ∩ valid	CodeGPT	– (no filter)	Figure 8
UniXcoder	¬EM ∪ ¬valid	UniXcoder	– (no filter)	Figure 9
UniXcoder	EM ∩ valid	UniXcoder	– (no filter)	Figure 10

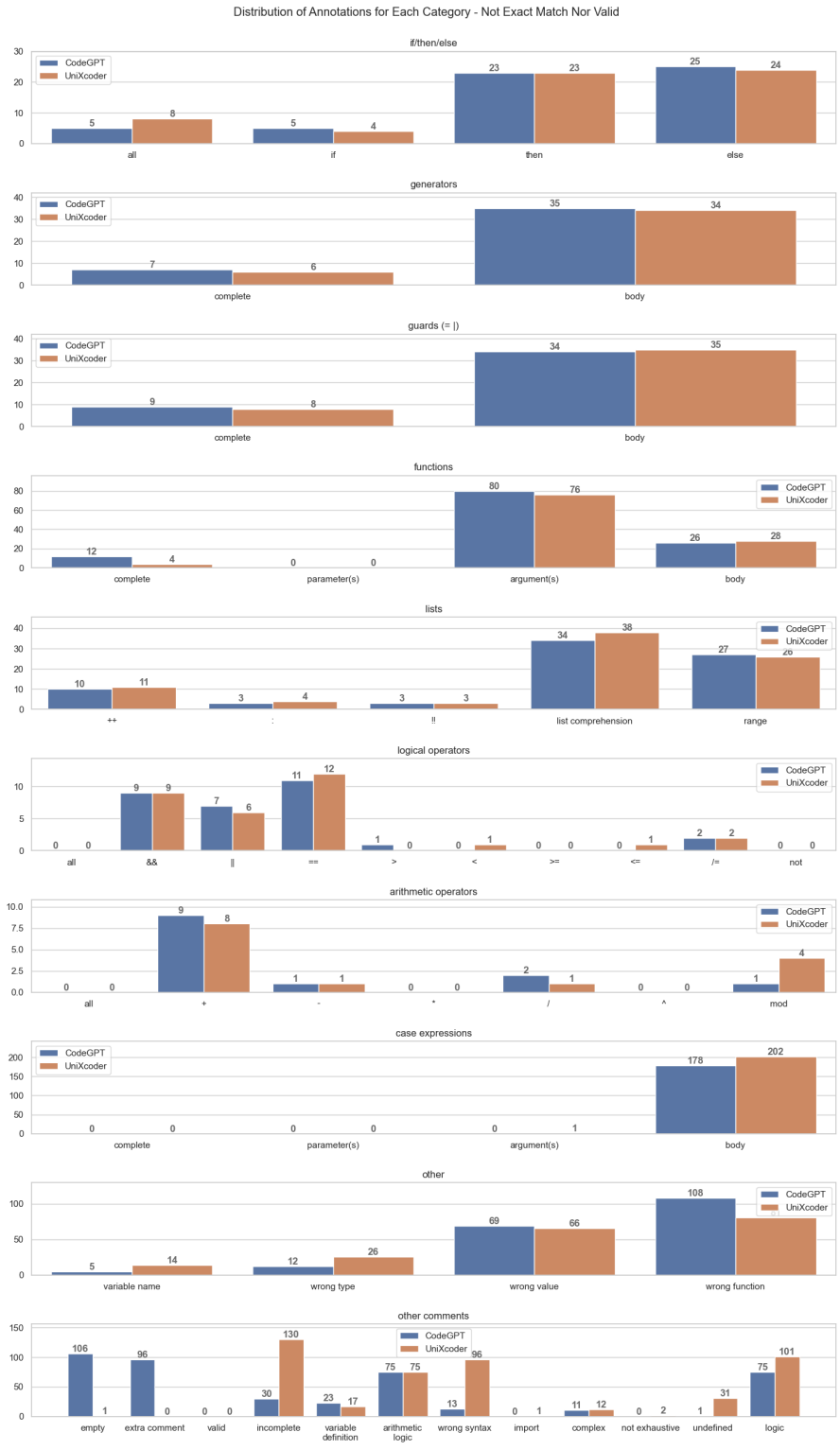

Figure 5: Distribution between CodeGPT and UniXcoder of manual annotations for samples that were neither an exact match (EM) nor deemed as valid.

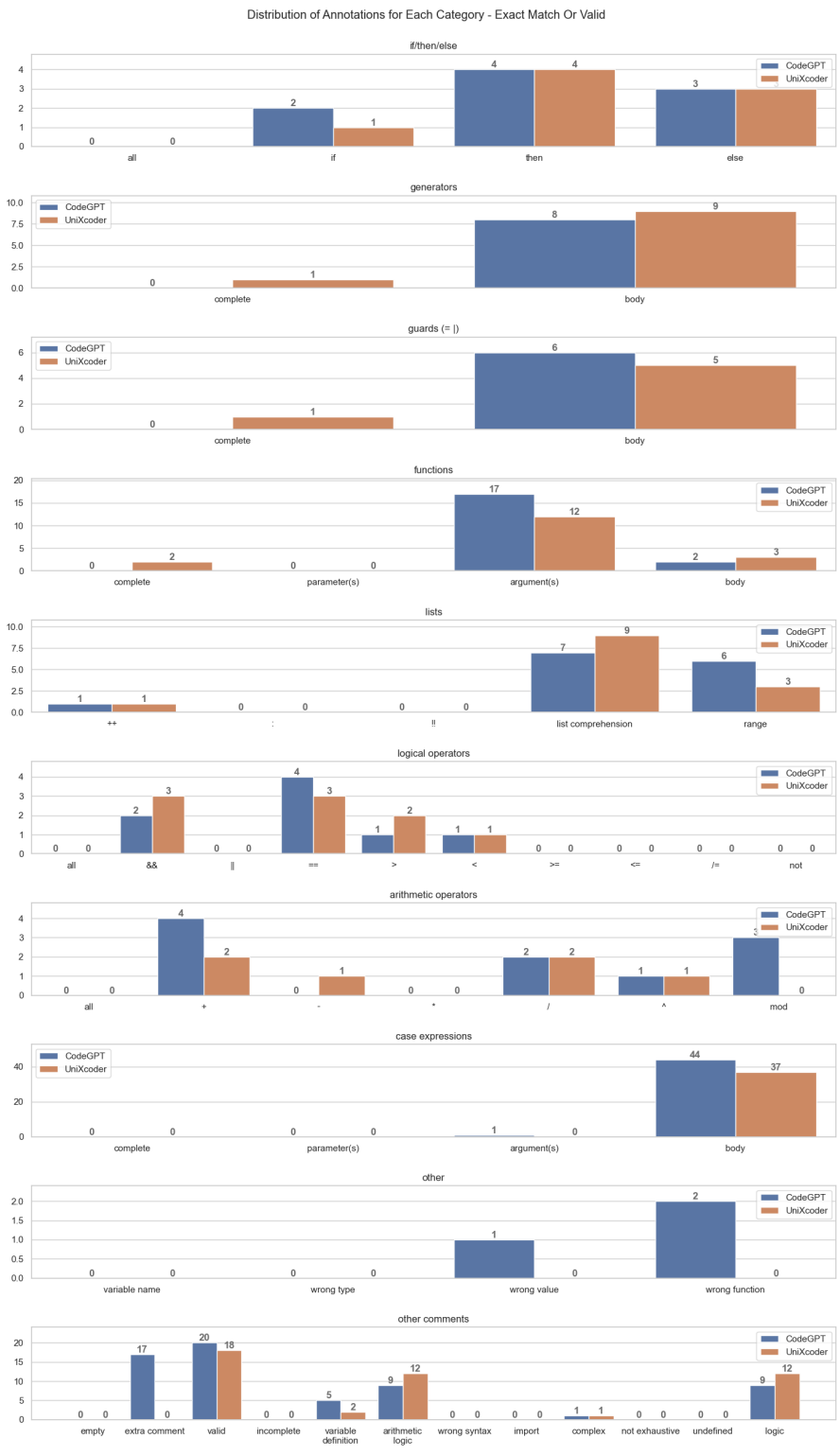

Figure 6: Distribution between CodeGPT and UniXcoder of manual annotations for samples that were either an exact match (EM) or deemed as valid.

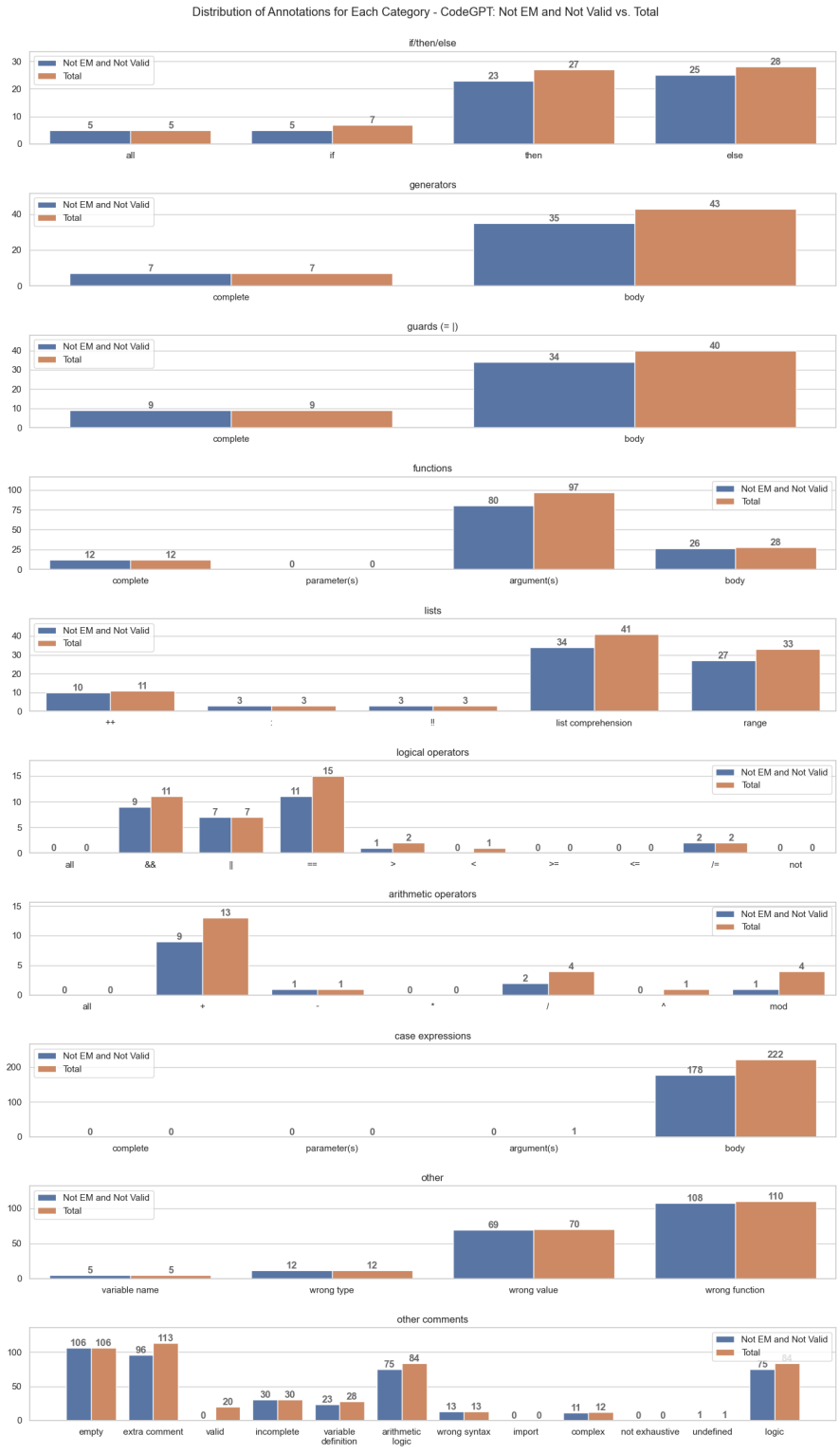

Figure 7: Distribution between CodeGPT's manual annotations for samples that were neither an exact match (EM) nor deemed as valid with respect to the total manual annotations made.

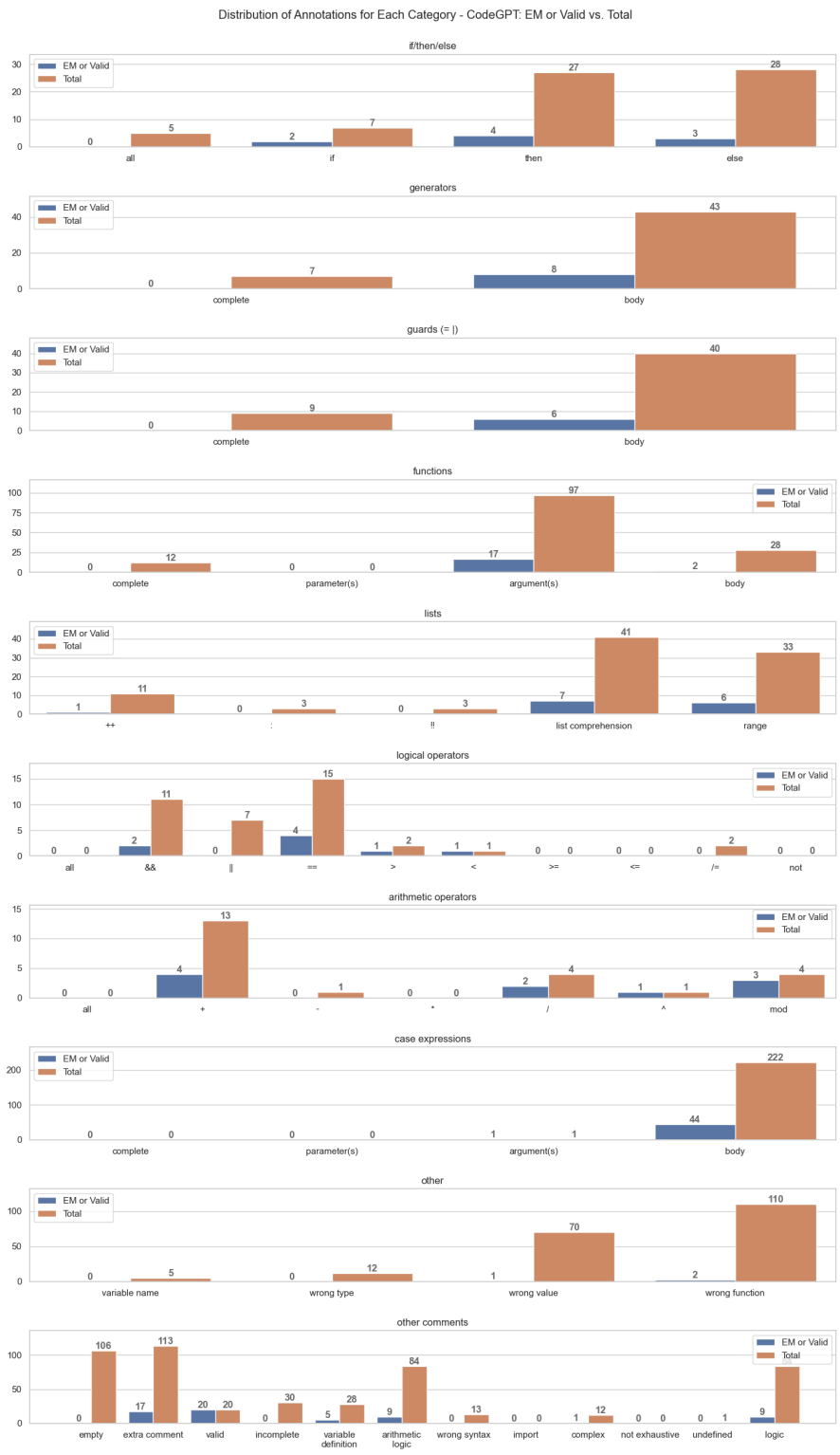

Figure 8: Distribution between CodeGPT's manual annotations for samples that were either an exact match (EM) or deemed as valid with respect to the total manual annotations made.

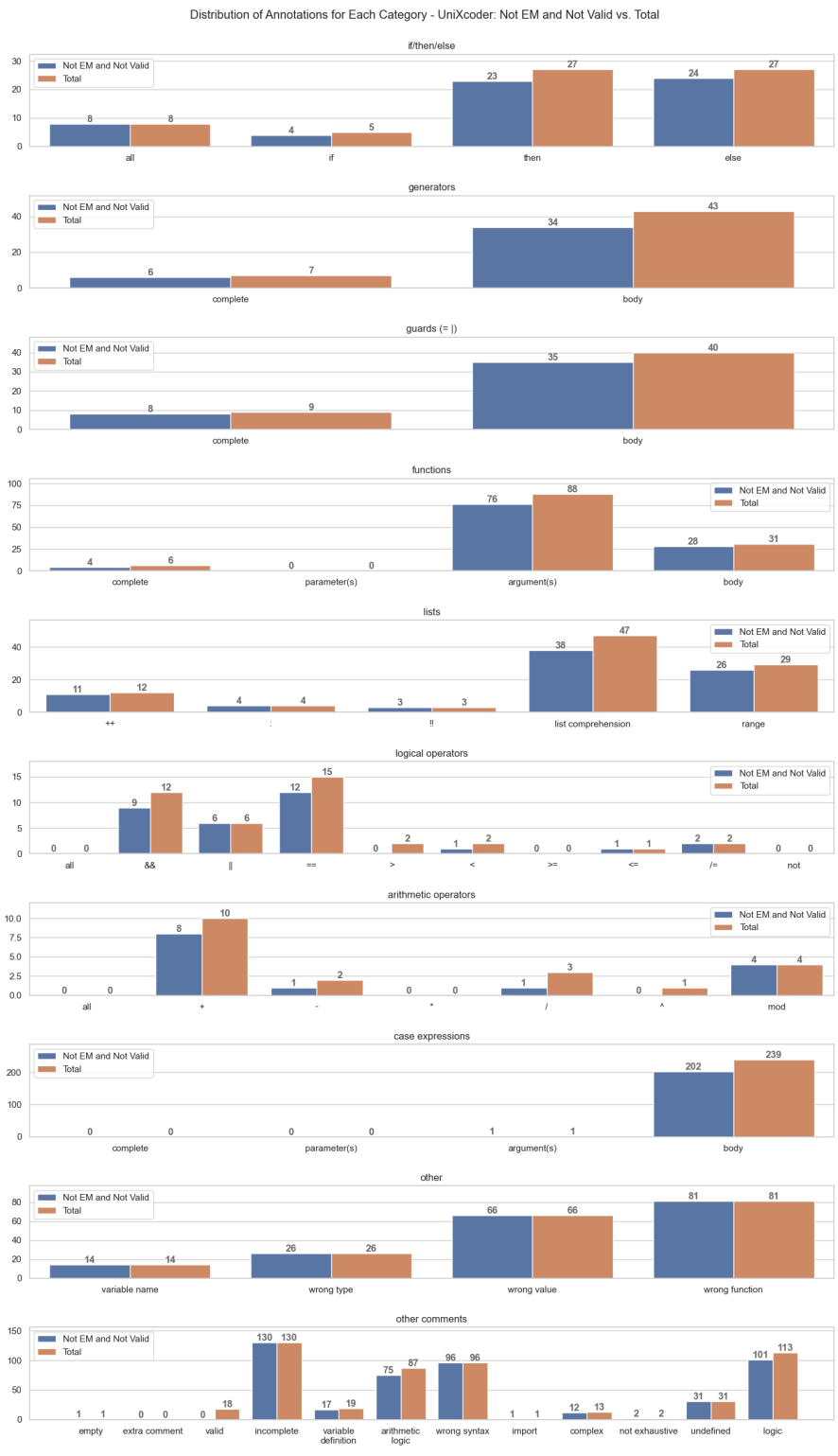

Figure 9: Distribution between UniXcoder's manual annotations for samples that were neither an exact match (EM) nor deemed as valid with respect to the total manual annotations made.

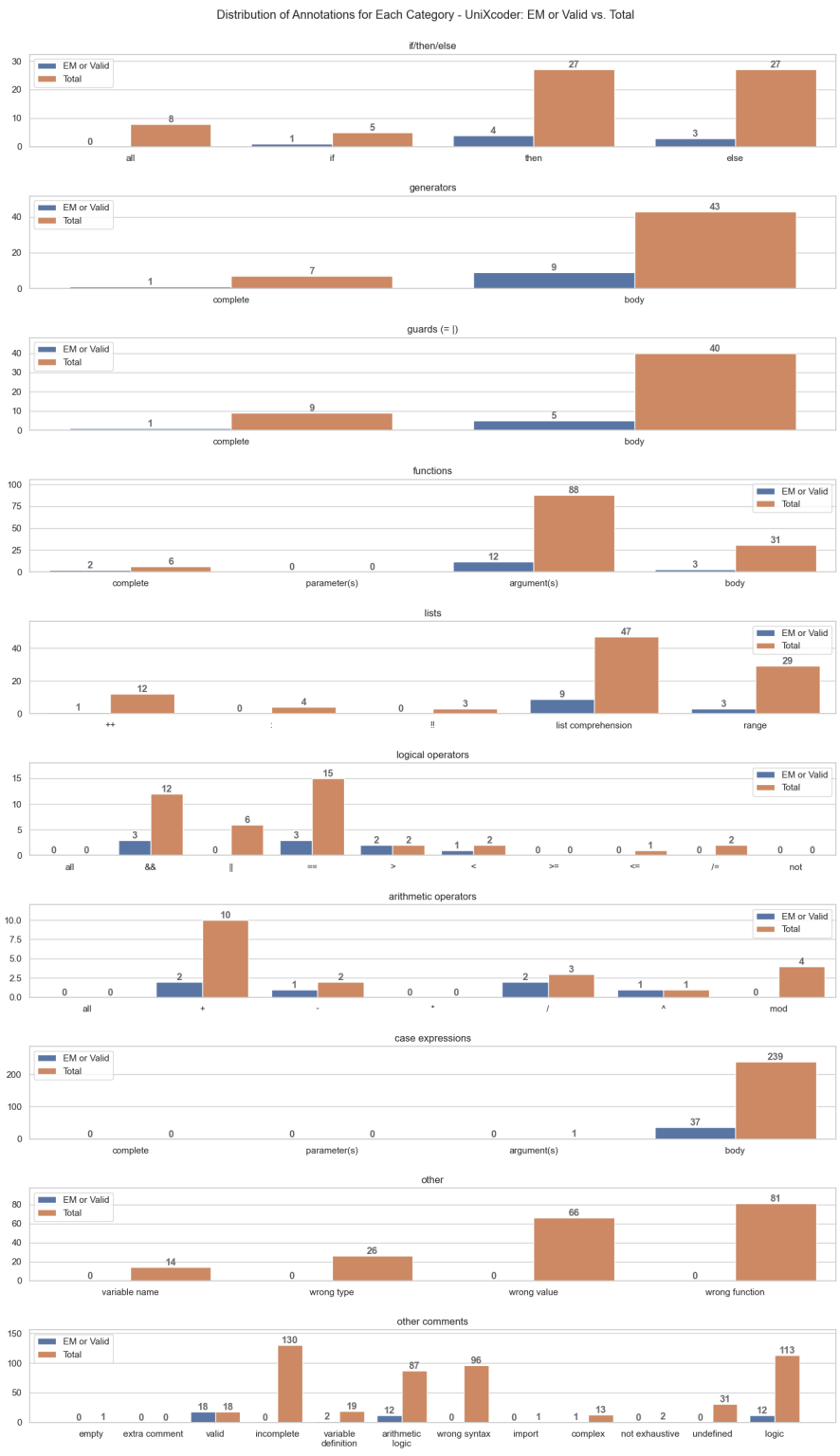

Figure 10: Distribution between UniXcoder's manual annotations for samples that were either an exact match (EM) or deemed as valid with respect to the total manual annotations made.