analysis_initial_golden

July 25, 2025

[1]:
```
import json
import re
import requests
import itertools
from collections import defaultdict, Counter
import pandas as pd
from itertools import combinations
import matplotlib.pyplot as plt
```

[2]:
```
with open("data/initial_dataset.json", "r") as f:
    initial_dataset = json.load(f)

with open("data/gold_standard.json", "r") as f:
    gold_standard_dataset = json.load(f)
```

[3]:
```
pwc_cat_cnt = []
orkg_cat_cnt = []
openalex_cat_cnt = []
openaire_cat_cnt = []
cat_cnt = []
cat_cnt2 = []
for paper in initial_dataset:
    if paper['openaire_categories_flat']:
        openaire_cat_cnt.append(len(paper['openaire_categories_flat']))
        cat_cnt.append(len(paper['openaire_categories_flat']))
    if paper['openalex_categories_flat']:
        openalex_cat_cnt.append(len(paper['openalex_categories_flat']))
        cat_cnt.append(len(paper['openalex_categories_flat']))
    if paper['papers_with_code_categories_flat']:
        pwc_cat_cnt.append(len(paper['papers_with_code_categories_flat']))
        cat_cnt.append(len(paper['papers_with_code_categories_flat']))
    if paper['orkg_categories_flat']:
        orkg_cat_cnt.append(len(paper['orkg_categories_flat']))
        cat_cnt.append(len(paper['orkg_categories_flat']))
    cat_cnt2.append(openaire_cat_cnt[-1] + openalex_cat_cnt[-1] +␣
 ↪pwc_cat_cnt[-1] + orkg_cat_cnt[-1])
```

```
print(f"avg cats initial: {round(sum(cat_cnt)/len(cat_cnt), 2)}")
print(f"avg2 cats initial: {round(sum(cat_cnt2)/len(cat_cnt2), 2)}")
print(f"sum cats initial: {sum(cat_cnt)}")
print(f'len cats initial: {len(cat_cnt)}')
print(f"OpenAlex cats initial: {sum(openalex_cat_cnt)}")
print(f"avg OpenAlex cats initial: {round(sum(openalex_cat_cnt)/
 ↪len(openalex_cat_cnt), 2)}")
print(f"OpenAIRE cats initial: {sum(openaire_cat_cnt)}")
print(f"avg OpenAIRE cats initial: {round(sum(openaire_cat_cnt)/
 ↪len(openaire_cat_cnt), 2)}")
print(f"with PwC cats initial: {sum(pwc_cat_cnt)}")
print(f"avg with PwC cats initial: {round(sum(pwc_cat_cnt)/len(pwc_cat_cnt),␣
 ↪2)}")
print(f"ORKG cat initial: {sum(orkg_cat_cnt)}")
print(f"avg ORKG cat initial: {round(sum(orkg_cat_cnt)/len(orkg_cat_cnt), 2)}")
```

```
avg cats initial: 9.84
avg2 cats initial: 39.37
sum cats initial: 2756
len cats initial: 280
OpenAlex cats initial: 867
avg OpenAlex cats initial: 12.39
OpenAIRE cats initial: 520
avg OpenAIRE cats initial: 7.43
with PwC cats initial: 1171
avg with PwC cats initial: 16.73
ORKG cat initial: 198
avg ORKG cat initial: 2.83
```

[4]:
```
pwc_cat_cnt = []
orkg_cat_cnt = []
openalex_cat_cnt = []
openaire_cat_cnt = []
cat_cnt = []
for paper in gold_standard_dataset:
    if paper['openaire_categories_flat']:
        openaire_cat_cnt.append(len(paper['openaire_categories_flat']))
        cat_cnt.append(len(paper['openaire_categories_flat']))
    if paper['openalex_categories_flat']:
        openalex_cat_cnt.append(len(paper['openalex_categories_flat']))
        cat_cnt.append(len(paper['openalex_categories_flat']))
    if paper['papers_with_code_categories_flat']:
        pwc_cat_cnt.append(len(paper['papers_with_code_categories_flat']))
        cat_cnt.append(len(paper['papers_with_code_categories_flat']))
    if paper['orkg_categories_flat']:
        orkg_cat_cnt.append(len(paper['orkg_categories_flat']))
        cat_cnt.append(len(paper['orkg_categories_flat']))
```

```
print(f"avg cats golden: {round(sum(cat_cnt)/len(cat_cnt), 2)}")
print(f"sum cats golden: {sum(cat_cnt)}")
print(f'len cats golden: {len(cat_cnt)}')
print(f"OpenAlex cats golden: {sum(openalex_cat_cnt)}")
print(f"avg OpenAlex cats golden: {round(sum(openalex_cat_cnt)/
 ↪len(openalex_cat_cnt), 2)}")
print(f"OpenAIRE cats golden: {sum(openaire_cat_cnt)}")
print(f"avg OpenAIRE cats golden: {round(sum(openaire_cat_cnt)/
 ↪len(openaire_cat_cnt), 2)}")
print(f"with PwC cats golden: {sum(pwc_cat_cnt)}")
print(f"avg with PwC cats golden: {round(sum(pwc_cat_cnt)/len(pwc_cat_cnt),␣
 ↪2)}")
print(f"ORKG cat golden: {sum(orkg_cat_cnt)}")
print(f"avg ORKG cat golden: {round(sum(orkg_cat_cnt)/len(orkg_cat_cnt), 2)}")
```

```
avg cats golden: 3.78
sum cats golden: 1046
len cats golden: 277
OpenAlex cats golden: 339
avg OpenAlex cats golden: 4.84
OpenAIRE cats golden: 257
avg OpenAIRE cats golden: 3.67
with PwC cats golden: 317
avg with PwC cats golden: 4.66
ORKG cat golden: 133
avg ORKG cat golden: 1.93
```

[5]:
```
categories = set([])
orkg_categories = set([])
pwc_categories = set([])
openalex_categories = set([])
openaire_categories = set([])

for paper in initial_dataset:
    for cat in paper['openaire_categories_flat']:
        categories.add(cat)
        openaire_categories.add(cat)
    for cat in paper['openalex_categories_flat']:
        categories.add(cat)
        openalex_categories.add(cat)
    for cat in paper['papers_with_code_categories_flat']:
        categories.add(cat)
        pwc_categories.add(cat)
    for cat in paper['orkg_categories_flat']:
        categories.add(cat)
        orkg_categories.add(cat)
```

```
print(f"#all categories cleaned: {len(list(categories))}")
print(f"#ORKG categories cleaned: {len(list(orkg_categories))}")
print(f"#PwC categories cleaned: {len(list(pwc_categories))}")
print(f"#OpenAlex categories cleaned: {len(list(openalex_categories))}")
print(f"#OpenAIRE categories cleaned: {len(list(openaire_categories))}")
```

```
#all categories cleaned: 728
#ORKG categories cleaned: 133
#PwC categories cleaned: 198
#OpenAlex categories cleaned: 277
#OpenAIRE categories cleaned: 157
```

[6]:
```
categories = set([])
orkg_categories = set([])
pwc_categories = set([])
openalex_categories = set([])
openaire_categories = set([])

for paper in gold_standard_dataset:
    for cat in paper['openaire_categories_flat']:
        categories.add(cat)
        openaire_categories.add(cat)
    for cat in paper['openalex_categories_flat']:
        categories.add(cat)
        openalex_categories.add(cat)
    for cat in paper['papers_with_code_categories_flat']:
        categories.add(cat)
        pwc_categories.add(cat)
    for cat in paper['orkg_categories_flat']:
        categories.add(cat)
        orkg_categories.add(cat)

print(f"#all categories golden: {len(list(categories))}")
print(f"#ORKG categories golden: {len(list(orkg_categories))}")
print(f"#PwC categories golden: {len(list(pwc_categories))}")
print(f"#OpenAlex categories golden: {len(list(openalex_categories))}")
print(f"#OpenAIRE categories golden: {len(list(openaire_categories))}")
```

```
#all categories golden: 300
#ORKG categories golden: 75
#PwC categories golden: 119
#OpenAlex categories golden: 96
#OpenAIRE categories golden: 38
```

[7]:
```
# Mapping of category to SKGs it's found in
category_to_skgs = defaultdict(set)
```

```
for paper in initial_dataset:
    skg_cats = {
        "orkg": set(paper["orkg_categories_flat"]),
        "pwc": set(paper["papers_with_code_categories_flat"]),
        "openalex": set(paper["openalex_categories_flat"]),
        "openaire": set(paper["openaire_categories_flat"]),
    }

    for skg, cats in skg_cats.items():
        for cat in cats:
            category_to_skgs[cat].add(skg)

# Count how many categories appear in 1, 2, 3, or 4 SKGs
agreement_counter = Counter()
for cat, skgs in category_to_skgs.items():
    agreement_counter[len(skgs)] += 1

# Total unique categories
total_unique_cats = len(category_to_skgs)

print(f"\nTotal unique categories: {total_unique_cats}")
print("\nAgreement levels:")
for k in range(1, 5):
    print(f"Categories appearing in {k} SKGs: {agreement_counter[k]}")
```

```
Total unique categories: 728

Agreement levels:
Categories appearing in 1 SKGs: 695
Categories appearing in 2 SKGs: 29
Categories appearing in 3 SKGs: 4
Categories appearing in 4 SKGs: 0
```

[8]:
```
# Mapping of category to SKGs it's found in
category_to_skgs = defaultdict(set)

for paper in gold_standard_dataset:
    skg_cats = {
        "orkg": set(paper["orkg_categories_flat"]),
        "pwc": set(paper["papers_with_code_categories_flat"]),
        "openalex": set(paper["openalex_categories_flat"]),
        "openaire": set(paper["openaire_categories_flat"]),
    }

    for skg, cats in skg_cats.items():
```

```
        for cat in cats:
            category_to_skgs[cat].add(skg)

# Count how many categories appear in 1, 2, 3, or 4 SKGs
agreement_counter = Counter()
for cat, skgs in category_to_skgs.items():
    agreement_counter[len(skgs)] += 1

# Total unique categories
total_unique_cats = len(category_to_skgs)

print(f"\nTotal unique categories: {total_unique_cats}")
print("\nAgreement levels:")
for k in range(1, 5):
    print(f"Categories appearing in {k} SKGs: {agreement_counter[k]}")
```

```
Total unique categories: 300

Agreement levels:
Categories appearing in 1 SKGs: 277
Categories appearing in 2 SKGs: 18
Categories appearing in 3 SKGs: 5
Categories appearing in 4 SKGs: 0
```

[9]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

skg_keys = {
    "pwc": "papers_with_code_categories_flat",
    "openalex": "openalex_categories_flat",
    "openaire": "openaire_categories_flat",
    "orkg": "orkg_categories_flat"
}

# Compute metrics per SKG
results = {}

for skg, key in skg_keys.items():
    y_true_all = []
    y_pred_all = []

    for paper_clean, paper_gold in zip(initial_dataset, gold_standard_dataset):
        gold_cats = set(cat.lower() for cat in paper_gold.get(key, []))
        pred_cats = set(cat.lower() for cat in paper_clean.get(key, []))

        all_cats = sorted(gold_cats | pred_cats)
        y_true = [1 if c in gold_cats else 0 for c in all_cats]
```

```
        y_pred = [1 if c in pred_cats else 0 for c in all_cats]

        y_true_all.extend(y_true)
        y_pred_all.extend(y_pred)

    precision = precision_score(y_true_all, y_pred_all, zero_division=0)
    recall = recall_score(y_true_all, y_pred_all, zero_division=0)
    f1 = f1_score(y_true_all, y_pred_all, zero_division=0)

    results[skg] = {
        "precision": round(precision, 2),
        "recall": round(recall, 2),
        "f1_score": round(f1, 2)
    }

# Output results
for skg, metrics in results.items():
    print(f"{skg.upper()}: Precision={metrics['precision']},␣
 ↪Recall={metrics['recall']}, F1-score={metrics['f1_score']}")
```

```
PWC: Precision=0.27, Recall=0.99, F1-score=0.42
OPENALEX: Precision=0.39, Recall=1.0, F1-score=0.56
OPENAIRE: Precision=0.35, Recall=0.72, F1-score=0.47
ORKG: Precision=0.65, Recall=0.97, F1-score=0.78
```

[10]:
```
gold_index = {paper["title"].lower(): paper for paper in gold_standard_dataset}

# SKGs and inconsistency counters
skg_keys = {
    "pwc": "papers_with_code_categories_flat",
    "orkg": "orkg_categories_flat",
    "openalex": "openalex_categories_flat",
    "openaire": "openaire_categories_flat"
}

inconsistencies = {
    skg: {
        "coverage_inconsistency": 0,
        "incorrect_assignment": 0,
    } for skg in skg_keys
}

# Go through each paper in cleaned data that is also in gold
for paper in initial_dataset:
    title = paper["title"].lower()
    if title not in gold_index:
        continue
```

```
    gold_paper = gold_index[title]

    for skg, skg_field in skg_keys.items():
        skg_labels = set(paper[skg_field])
        gold_labels = set(gold_paper[skg_field])

        # Coverage inconsistency
        if len(skg_labels) <= 1 and len(gold_labels) >= 3:
            inconsistencies[skg]["coverage_inconsistency"] += 1

        # Incorrect assignment (labels not in gold)
        if len(skg_labels - gold_labels) > 0:
            inconsistencies[skg]["incorrect_assignment"] += 1

# Output results
print("Inconsistency counts per SKG:")
for skg in skg_keys:
    print(f"{skg.upper()}:␣
 ↪Coverage={inconsistencies[skg]['coverage_inconsistency']}, "
          f"Incorrect={inconsistencies[skg]['incorrect_assignment']}")
```

```
Inconsistency counts per SKG:
PWC: Coverage=0, Incorrect=54
ORKG: Coverage=0, Incorrect=30
OPENALEX: Coverage=0, Incorrect=60
OPENAIRE: Coverage=0, Incorrect=61
```

[11]:
```
gold_index = {paper["title"].lower(): paper for paper in gold_standard_dataset}
skg_keys = {
    "pwc": "papers_with_code_categories_flat",
    "orkg": "orkg_categories_flat",
    "openalex": "openalex_categories_flat",
    "openaire": "openaire_categories_flat"
}

extra_counts = defaultdict(int)
paper_counts = defaultdict(int)
initial_total = defaultdict(int)
gold_total = defaultdict(int)

for paper in initial_dataset:
    title = paper["title"].lower()
    if title not in gold_index:
        continue
    gold_paper = gold_index[title]

    for skg, field in skg_keys.items():
```

```
        cleaned_labels = set(paper[field])
        gold_labels = set(gold_paper[field])
        extras = cleaned_labels - gold_labels

        extra_counts[skg] += len(extras)
        paper_counts[skg] += 1
        initial_total[skg] += len(cleaned_labels)
        gold_total[skg] += len(gold_labels)

# Print results
print("SKG - Initial - Gold - Incorrect - AvgIncorrect")
for skg in skg_keys:
    total_init = initial_total[skg]
    total_gold = gold_total[skg]
    total_extra = extra_counts[skg]
    avg_extra = total_extra / paper_counts[skg]
    print(f"{skg.upper()} - {total_init} - {total_gold} - {total_extra} -␣
 ↪{avg_extra:.2f}")
```

```
SKG - Initial - Gold - Incorrect - AvgIncorrect
PWC - 1018 - 243 - 779 - 12.56
ORKG - 184 - 123 - 65 - 1.05
OPENALEX - 801 - 311 - 490 - 7.90
OPENAIRE - 479 - 228 - 317 - 5.11
```

[12]:
```
from itertools import combinations

# SKG keys
skg_keys = {
    "PwC": "papers_with_code_categories_flat",
    "OpenAlex": "openalex_categories_flat",
    "OpenAIRE": "openaire_categories_flat",
    "ORKG": "orkg_categories_flat"
}

# Initialize storage
pairwise_totals = {pair: 0 for pair in combinations(skg_keys.keys(), 2)}
triple_totals = {triplet: 0 for triplet in combinations(skg_keys.keys(), 3)}
full_agreement_count = 0
num_papers = len(initial_dataset)

for paper in initial_dataset:
    skg_cats = {skg: set(paper.get(key, [])) for skg, key in skg_keys.items()}

    # Pairwise overlaps
    for skg1, skg2 in pairwise_totals:
        overlap = skg_cats[skg1].intersection(skg_cats[skg2])
```

```
        pairwise_totals[(skg1, skg2)] += len(overlap)

    # Triple overlaps
    for skg1, skg2, skg3 in triple_totals:
        overlap = skg_cats[skg1] & skg_cats[skg2] & skg_cats[skg3]
        triple_totals[(skg1, skg2, skg3)] += len(overlap)

    # Full agreement across all SKGs
    all_overlap = set.intersection(*skg_cats.values())
    if all_overlap:
        full_agreement_count += 1

# Print average pairwise overlaps
print("Average pairwise overlaps per paper:")
for pair, total in pairwise_totals.items():
    print(f"{pair[0]} & {pair[1]}: {total}")

# Print average triple overlaps
print("\nAverage triple overlaps per paper:")
for triplet, total in triple_totals.items():
    print(f"{triplet[0]} & {triplet[1]} & {triplet[2]}: {total}")

# Full agreement
print(f"\nNumber of papers with full SKG overlap: {full_agreement_count} out of␣
 ↪{num_papers}")
```

```
Average pairwise overlaps per paper:
PwC & OpenAlex: 19
PwC & OpenAIRE: 3
PwC & ORKG: 8
OpenAlex & OpenAIRE: 29
OpenAlex & ORKG: 3
OpenAIRE & ORKG: 0

Average triple overlaps per paper:
PwC & OpenAlex & OpenAIRE: 0
PwC & OpenAlex & ORKG: 1
PwC & OpenAIRE & ORKG: 0
OpenAlex & OpenAIRE & ORKG: 0

Number of papers with full SKG overlap: 0 out of 70
```

[13]:
```
from itertools import combinations

# SKG keys
skg_keys = {
    "PwC": "papers_with_code_categories_flat",
```

```
    "OpenAlex": "openalex_categories_flat",
    "OpenAIRE": "openaire_categories_flat",
    "ORKG": "orkg_categories_flat"
}

# Initialize storage
pairwise_totals = {pair: 0 for pair in combinations(skg_keys.keys(), 2)}
triple_totals = {triplet: 0 for triplet in combinations(skg_keys.keys(), 3)}
full_agreement_count = 0
num_papers = len(gold_standard_dataset)

for paper in gold_standard_dataset:
    skg_cats = {skg: set(paper.get(key, [])) for skg, key in skg_keys.items()}

    # Pairwise overlaps
    for skg1, skg2 in pairwise_totals:
        overlap = skg_cats[skg1].intersection(skg_cats[skg2])
        pairwise_totals[(skg1, skg2)] += len(overlap)

    # Triple overlaps
    for skg1, skg2, skg3 in triple_totals:
        overlap = skg_cats[skg1] & skg_cats[skg2] & skg_cats[skg3]
        triple_totals[(skg1, skg2, skg3)] += len(overlap)

    # Full agreement across all SKGs
    all_overlap = set.intersection(*skg_cats.values())
    if all_overlap:
        full_agreement_count += 1

# Print average pairwise overlaps
print("Average pairwise overlaps per paper:")
for pair, total in pairwise_totals.items():
    print(f"{pair[0]} & {pair[1]}: {total}")

# Print average triple overlaps
print("\nAverage triple overlaps per paper:")
for triplet, total in triple_totals.items():
    print(f"{triplet[0]} & {triplet[1]} & {triplet[2]}: {total}")

# Full agreement
print(f"\nNumber of papers with full SKG overlap: {full_agreement_count} out of␣
 ↪{num_papers}")
```

```
Average pairwise overlaps per paper:
PwC & OpenAlex: 18
PwC & OpenAIRE: 4
PwC & ORKG: 7
```

```
OpenAlex & OpenAIRE: 71
OpenAlex & ORKG: 3
OpenAIRE & ORKG: 1

Average triple overlaps per paper:
PwC & OpenAlex & OpenAIRE: 0
PwC & OpenAlex & ORKG: 1
PwC & OpenAIRE & ORKG: 0
OpenAlex & OpenAIRE & ORKG: 1

Number of papers with full SKG overlap: 0 out of 70
```