detect_and_fix_dangles.py

```
import geopandas as gpd
from shapely.geometry import Point
from shapely.ops import snap

# Load shapefile
gdf = gpd.read_file("data/roads_with_errors.shp")

# Detect dangling lines
def detect_dangles(gdf):
    dangles = []
    for idx, geom in gdf.geometry.items():
        if geom.geom_type == 'LineString':
            start, end = Point(geom.coords[0]), Point(geom.coords[-1])
            connected = sum(
                any(endpt.intersects(other) for endpt in [start, end])
                for other in gdf.geometry if other != geom
            )
            if connected < 2:
                dangles.append(idx)
    return gdf.loc[dangles]

# Auto-correct dangles
def correct_dangles(gdf, threshold=1.0):
    corrected = [snap(geom, gdf.unary_union, threshold) for geom in gdf.geometry]
    gdf.geometry = corrected
    return gdf

# Run
dangles = detect_dangles(gdf)
print(f"Dangles found: {len(dangles)}")
gdf_fixed = correct_dangles(gdf)
gdf_fixed.to_file("data/roads_corrected.shp")
```

```
Dangles found: 4
<ipython-input-13-d45b36c33725>:24: DeprecationWarning: The 'unary_union' attribute is deprecated, use the 'union_all()' method ins
  corrected = [snap(geom, gdf.unary_union, threshold) for geom in gdf.geometry]
```

detect_sliver_polygons.py

```
import geopandas as gpd

# Load polygon layer
gdf = gpd.read_file("data/parcels_with_errors.shp")

# Detect sliver polygons by area threshold
def detect_slivers(gdf, area_threshold=5.0):
    slivers = gdf[gdf.area < area_threshold]
    return slivers

slivers = detect_slivers(gdf)
print(f"Sliver polygons found: {len(slivers)}")
slivers.to_file("data/slivers_detected.shp")
```

```
Sliver polygons found: 2
/usr/local/lib/python3.11/dist-packages/pyogrio/geopandas.py:662: UserWarning: 'crs' was not provided.  The output dataset will not
  write(
```

validate_topology.py

```
import geopandas as gpd

# Load polygon layer
gdf = gpd.read_file("data/parcels_with_errors.shp")

# Check for overlaps
def check_overlaps(gdf):
    overlaps = []
    for idx, row in gdf.iterrows():
        geom = row.geometry
        others = gdf.drop(index=idx)
        for _, other in others.iterrows():
            if geom.intersects(other.geometry) and not geom.touches(other.geometry):
                overlaps.append((idx, other.name))
    return overlaps

violations = check_overlaps(gdf)
print(f"Overlapping pairs: {len(violations)}")
```

```
Overlapping pairs: 0
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.