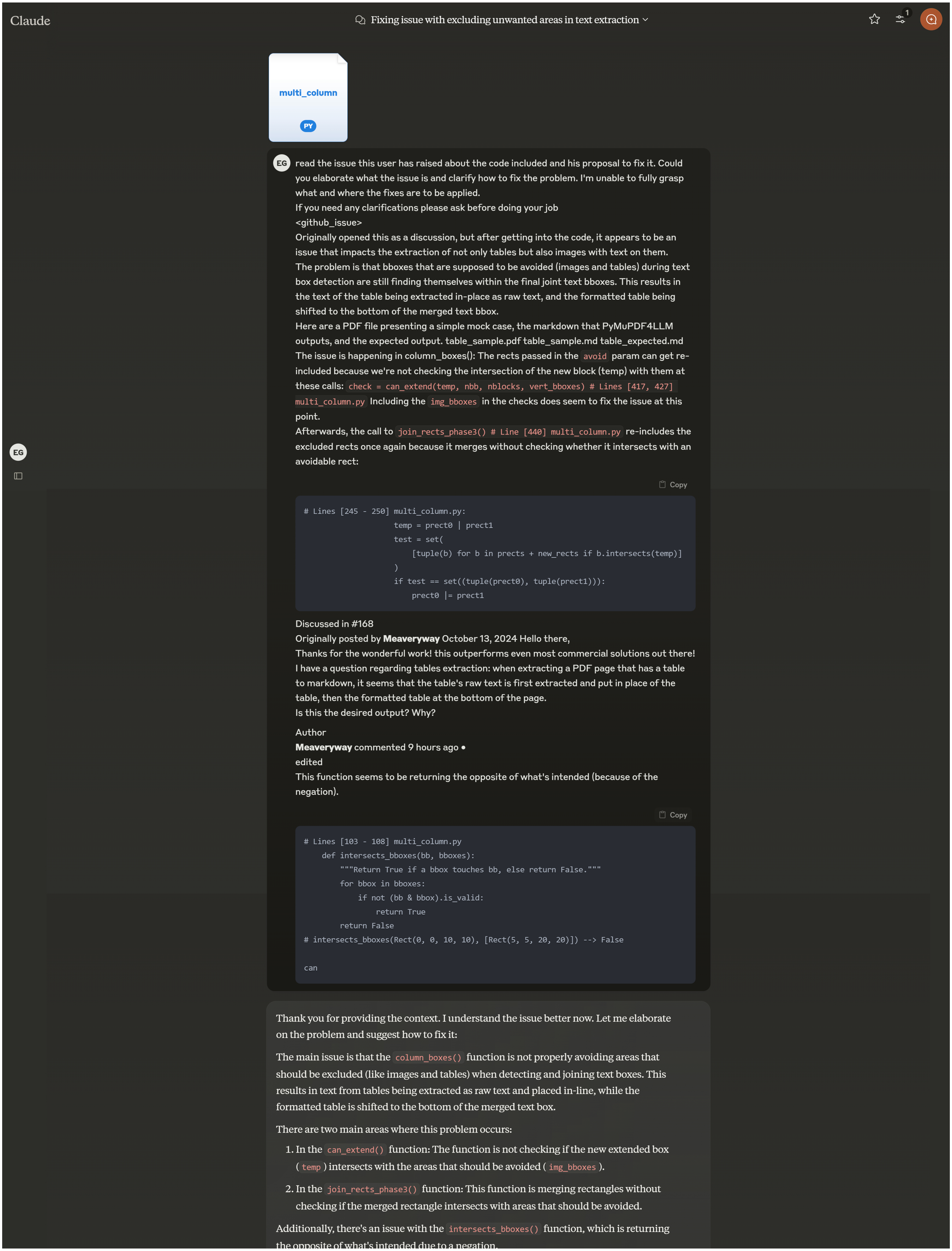

Claude
Fixing issue with excluding unwanted areas in text extraction
1
multi_column
PY
EG
EG

read the issue this user has raised about the code included and his proposal to fix it. Could you elaborate what the issue is and clarify how to fix the problem. I'm unable to fully grasp what and where the fixes are to be applied.
If you need any clarifications please ask before doing your job
<github_issue>
Originally opened this as a discussion, but after getting into the code, it appears to be an issue that impacts the extraction of not only tables but also images with text on them.
The problem is that bboxes that are supposed to be avoided (images and tables) during text box detection are still finding themselves within the final joint text bboxes. This results in the text of the table being extracted in-place as raw text, and the formatted table being shifted to the bottom of the merged text bbox.
Here are a PDF file presenting a simple mock case, the markdown that PyMuPDF4LLM outputs, and the expected output. table_sample.pdf table_sample.md table_expected.md
The issue is happening in column_boxes(): The rects passed in the `avoid` param can get re-included because we're not checking the intersection of the new block (temp) with them at these calls: `check = can_extend(temp, nbb, nblocks, vert_bboxes) # Lines [417, 427]` `multi_column.py` Including the `img_bboxes` in the checks does seem to fix the issue at this point.
Afterwards, the call to `join_rects_phase3() # Line [440] multi_column.py` re-includes the excluded rects once again because it merges without checking whether it intersects with an avoidable rect:

```
# Lines [245 - 250] multi_column.py:
                    temp = prect0 | prect1
                    test = set(
                        [tuple(b) for b in prects + new_rects if b.intersects(temp)]
                    )
                    if test == set((tuple(prect0), tuple(prect1))):
                        prect0 |= prect1
```

Discussed in #168
Originally posted by **Meaveryway** October 13, 2024 Hello there,
Thanks for the wonderful work! this outperforms even most commercial solutions out there!
I have a question regarding tables extraction: when extracting a PDF page that has a table to markdown, it seems that the table's raw text is first extracted and put in place of the table, then the formatted table at the bottom of the page.
Is this the desired output? Why?

Author
Meaveryway commented 9 hours ago •
edited
This function seems to be returning the opposite of what's intended (because of the negation).

```
# Lines [103 - 108] multi_column.py
    def intersects_bboxes(bb, bboxes):
        """Return True if a bbox touches bb, else return False."""
        for bbox in bboxes:
            if not (bb & bbox).is_valid:
                return True
        return False
# intersects_bboxes(Rect(0, 0, 10, 10), [Rect(5, 5, 20, 20)]) --> False

can
```

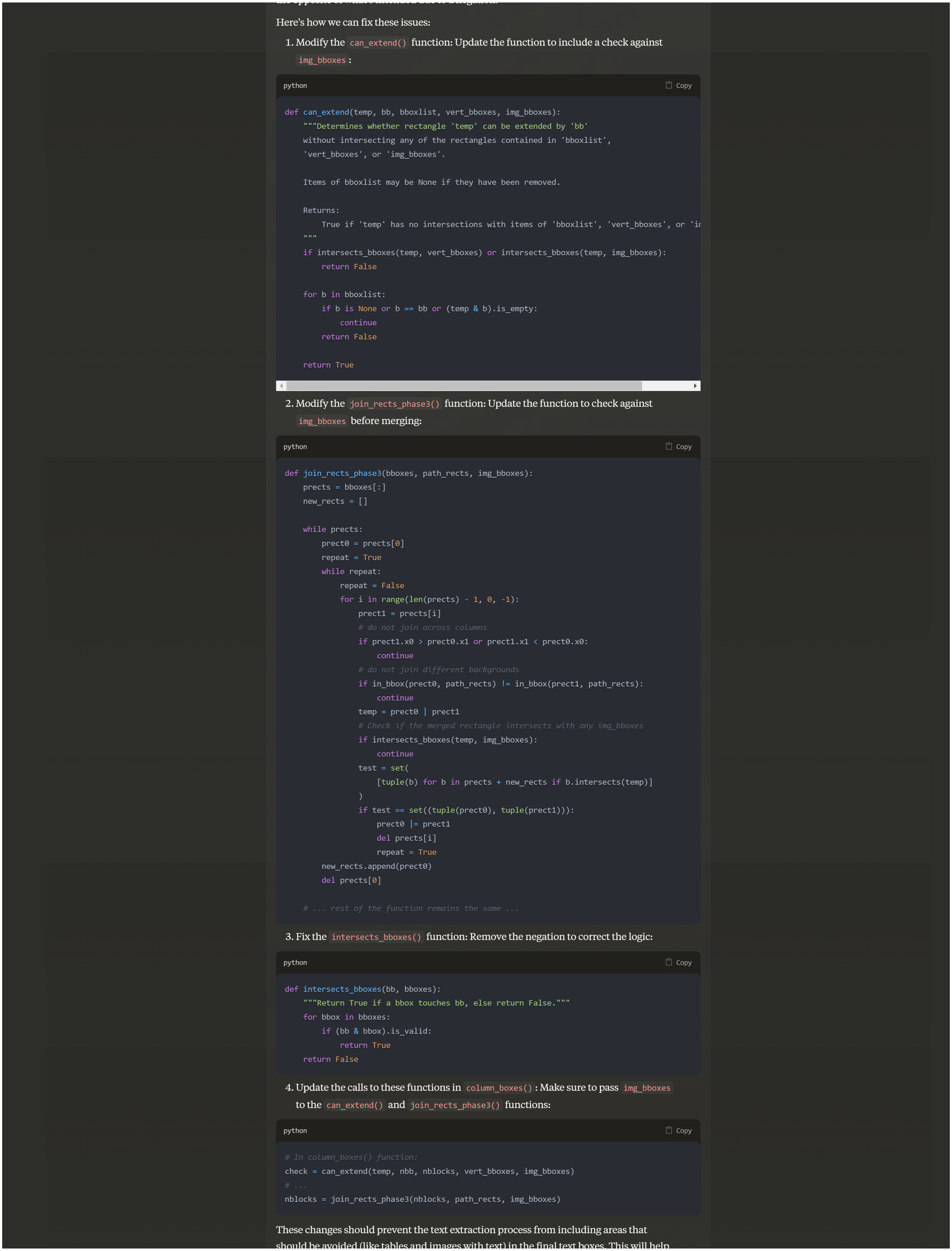
Thank you for providing the context. I understand the issue better now. Let me elaborate on the problem and suggest how to fix it:

The main issue is that the `column_boxes()` function is not properly avoiding areas that should be excluded (like images and tables) when detecting and joining text boxes. This results in text from tables being extracted as raw text and placed in-line, while the formatted table is shifted to the bottom of the merged text box.

There are two main areas where this problem occurs:

1. In the `can_extend()` function: The function is not checking if the new extended box (`temp`) intersects with the areas that should be avoided (`img_bboxes`).
2. In the `join_rects_phase3()` function: This function is merging rectangles without checking if the merged rectangle intersects with areas that should be avoided.

Additionally, there's an issue with the `intersects_bboxes()` function, which is returning the opposite of what's intended due to a negation.



Here's how we can fix these issues:

1. Modify the `can_extend()` function: Update the function to include a check against `img_bboxes`:

python Copy

```python
def can_extend(temp, bb, bboxlist, vert_bboxes, img_bboxes):
    """Determines whether rectangle 'temp' can be extended by 'bb'
    without intersecting any of the rectangles contained in 'bboxlist',
    'vert_bboxes', or 'img_bboxes'.

    Items of bboxlist may be None if they have been removed.

    Returns:
        True if 'temp' has no intersections with items of 'bboxlist', 'vert_bboxes', or 'i
    """
    if intersects_bboxes(temp, vert_bboxes) or intersects_bboxes(temp, img_bboxes):
        return False

    for b in bboxlist:
        if b is None or b == bb or (temp & b).is_empty:
            continue
        return False

    return True
```

2. Modify the `join_rects_phase3()` function: Update the function to check against `img_bboxes` before merging:

python Copy

```python
def join_rects_phase3(bboxes, path_rects, img_bboxes):
    prects = bboxes[:]
    new_rects = []

    while prects:
        prect0 = prects[0]
        repeat = True
        while repeat:
            repeat = False
            for i in range(len(prects) - 1, 0, -1):
                prect1 = prects[i]
                # do not join across columns
                if prect1.x0 > prect0.x1 or prect1.x1 < prect0.x0:
                    continue
                # do not join different backgrounds
                if in_bbox(prect0, path_rects) != in_bbox(prect1, path_rects):
                    continue
                temp = prect0 | prect1
                # Check if the merged rectangle intersects with any img_bboxes
                if intersects_bboxes(temp, img_bboxes):
                    continue
                test = set(
                    [tuple(b) for b in prects + new_rects if b.intersects(temp)]
                )
                if test == set((tuple(prect0), tuple(prect1))):
                    prect0 |= prect1
                    del prects[i]
                    repeat = True
        new_rects.append(prect0)
        del prects[0]

    # ... rest of the function remains the same ...
```

3. Fix the `intersects_bboxes()` function: Remove the negation to correct the logic:

python Copy

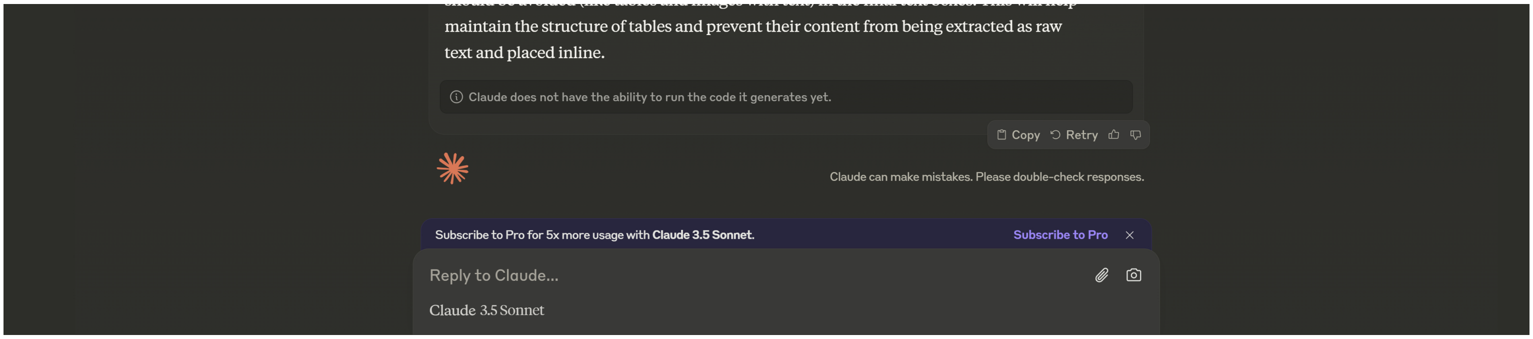
```python
def intersects_bboxes(bb, bboxes):
    """Return True if a bbox touches bb, else return False."""
    for bbox in bboxes:
        if (bb & bbox).is_valid:
            return True
    return False
```

4. Update the calls to these functions in `column_boxes()`: Make sure to pass `img_bboxes` to the `can_extend()` and `join_rects_phase3()` functions:

python Copy

```python
# In column_boxes() function:
check = can_extend(temp, nbb, nblocks, vert_bboxes, img_bboxes)
# ...
nblocks = join_rects_phase3(nblocks, path_rects, img_bboxes)
```

These changes should prevent the text extraction process from including areas that should be avoided (like tables and images with text) in the final text boxes. This will help



maintain the structure of tables and prevent their content from being extracted as raw text and placed inline.

Claude does not have the ability to run the code it generates yet.

Claude can make mistakes. Please double-check responses.

Subscribe to Pro for 5x more usage with **Claude 3.5 Sonnet.**