

Here is just some straight up text, no math or equations or anything. I just want to see if I can visualize the information that is output to `boxpositions_{filename}.txt` without writing a huge file and drawing a bunch of rectangles for all the different words. I'll also need to think a little more clearly on how exactly I intend to use all the information in that `boxpositions_{filename}.txt` file. Let's do that now—why not. So in theory the information this gives me is

- (1) the width, height, and depth of words in the source that are not inside forbidden environments on a naked block and are surrounded by just space, no punctuation.
- (2) the start and end x and y coordinate of the word in the PDF: I should be able to use this to get the word's x_0 and x_1 rectangle in the PDF
- (3) the start and end page of the word in the PDF

I believe the start and end y coordinates are always the same unless the word spans a newline or pagebreak, which might be worth using? Or maybe I'll just ignore those? Ultimately I want some function f where $f(\text{rectangle in PDF}) = \text{the source LATEX which created and/or is very near that rectangle}$. So....

Okay! This might be overkill, but we could do something like this: Given an arbitrary page and rectangle specified by x_0, y_0, x_1, y_1 see which individual word rectangles on that page the larger rectangle intersects.

Then identify the intersecting word rectangle that is earliest in the page (smallest y_0 then x_0 —I believe a larger y coordinate means further down the page; it's not the usual cartesian coordinate system) and the one which is last in the page (largest y_1 and then largest x_1) and then just grab all of the LATEX source which is between the first and last and call it a day—with the exception of figure source.

If a rectangle is given which selects the caption of a figure that shouldn't¹ intersect any word rectangles because only words which are in the body or enunciations or proofs or the bibliography are marked. So ... nice.

Right now the next thing I need to do is, though, actually create those word (and actually also inline math, now that I think of it) rectangles and draw them and see if they're accurate.

¹(fingers crossed)