File-Names and Summary

File-Names

- *_MAIN.html,* this is the code solution with relevant comments.
- *NER-Rental* is the model name.
- *OutputCSVs*, the directory with the required output.
  - (*Training_convert_labels* and *Validation_convert_labels*)
- *Training_data*, is the directory with docx converted training files, the solution has been implemented for PDFs (easily switchable).
- *Validation_Data*, is the directory with docx converted validation files, the solution has been implemented for PDFs (easily switchable).
- *PDFs*, the converted PDFs are outputted to this folder.
- *Screenshots,* this folder contains compiled screenshots of annotation tools in progress and model results.
- *main-zy-anno,* is the final actual notebook.
- *annotations.csv,* this is the manually annotated output.

- Since no regex, next step is getting the model to recognise entities.
- Entity recognition is only possible through model if the model has seen examples.
- We have PDFs which are converted to docx for assignment purposes.
- Spacy requires a list of tuples.
- Wrote a function to convert all docx back to their original forms [PDFs as intended] and next step is annotation.
- **Doccano, Inception, NER-annotator, Prodigy, Label-Studio.**
- Doccano cannot input PDF and it's text parsing doesnt support overflow.
- Doccano and Label studio cannot output to CONLL or UIMA CAS JSON, so that is another drawback.
- NER-Annotator, does have the requisite format for spacy but it only takes text as input which is a lot of manual work.
- Inception is JAR-based application required Java 11 and uses active learning like Prodigy.
- Prodigy is paid and is therefore not suitable.
- All of the above tools work through a local server deployed on the client machine
- Inception works the best for this use-case due to the fact that it can output a lot of useful formats. They are listed below.
- Inception outputs :
  - CONLL 2002
  - CONLL 2003

    - CONLL 2012 ETC
    - CONLL NLP
    - CONLLU
    - UIMA CAS JSON
    - WEBANNO
- Wrote functions for all of the above, none worked.
- Tried to parse docx to tsv, tsv to JSON, JSON to tuple, the format didn’t match
- OCR through teserract can be discarded since the PDFs are parseable. Also the reason docs were converted back to PDFs was because of encoding. MS encoding is different from PDF encoding and some PDFs might display garbled hex chars when imported as docx.
- Since all PDF formats are different, custom labels are necessary.
- So used, a **custom library** called “spacy-annotator” to manually annotate through Jupyter widgets, works fantastically well for the use-case. [the desired format is a list of tuples, containing a string (the text) and a dictionary. The dictionary has one entry 'entities', and its value is a list of tuples (triples) consisting of two integers (the start and end index) and a string (the label)]
- Other problems with custom labels without external libraries :
    - Sentence splits are done based on largely different approaches. [Tried, NLTK and Spacy tokenizer, Sentencesplitter custom function, Regex split and String split on each text.]
    - So with all approaches the sentence splits are different, some output 42 some out 30 some output 16. Which is understood [for example: S.M.Ravichandran, a split here itself gives us 3/2 splits depending on method, but we do not want to split on that.]
    - The BIAT approach, the one that Inception uses cannot be replicated through a script.