Overview

Objective of this notebook is 4-folds -

1. **Assemble / Generate** meeting transcripts data. (obvious needs of preprocessing and cleaning with a little EDA)
2. Using some example outputs - **form heuristics** (based on NLP features of the interested input)
3. **Validate the heuristics** -- how well does the heuristic extract necessary items? (accordingly repeat this step and the previous till desired results comeup)
4. Formalize the operations to a single function for **service intergration**

Evee-Output

Calling out my assumptions of what can be a good response from the NLP service.
From the transcription, I am considering following information to be of interest for a meeting MoM.

Content of interest in input

1. **Directly Address Evee**

   - `Hey Evee, {can you | please  | can you please} add  (task) X to person (Y)`
   - *Just a sample. Same sentence can be said in various ways*
     - `Evee, {can you | please | ... } put (task) X to person (Y)'s list`
     - `Remember to assign (task) X to (person) Y, Evee`
   - Above is easier to implement as we can narrow down to those part of transcripts where Evee is adressed

1. **Actionable or Commitable Item phrases**

   - Dialogues with possible action items
     `I have been doing this but will update you with other alternatives`
   - **Note:-** Action items can be part of a previous reference

     ```
      05:10 --> 05:12 : I was looking into (task) X and I did action A.
     05:13 --> 05:16 : True (person name) Y, but I feel we should also try other (ac
     tion) B. Can (you| person Y) do it?
     05:16 --> 05:17 : Sure
     ```

   - Once the actions are identified, map with the person's involved
     - Involved person may be already mentioned in the tasks
     - Check for phrases of `I|You` etc.

**Note: I will assume there can be false positives of tasks generated although it wasn't decided to go ahead with it..
Overall, more tasks generated is better than less tasks**

Also, assumption is if the ownership of a task changes we can add both previous and the new user and let it be modifiable through actionable cards

Service Output

```
## list of todos
results = [{actionEntity  , intent , peopleEntity, MoMString}, ....]
```

where
actionEntity = token(s) describing the tasks

intent = verb - Design/Develop etc.

peopleEntity = names list of persons involved

MoMString = generated from above three to be directly used in adaptive card

Data assemble

In [12]:
```
## Taking very simple static strings as text for now -
## In hack, we need a vtt file parser to extract text from therein
## For example,

"""
0:0:0.0 --> 0:0:5.290
<v 1ae37f18-c4fb-454a-8514-ad5b764b92f1@72f988bf-86f1-41af-91ab-2d7cd011db47>Yea
h, I think please don't read the transcript because this transcripts for my Engli
sh especially is very bad so.</v>

0:0:9.250 --> 0:0:11.520
<v 1ae37f18-c4fb-454a-8514-ad5b764b92f1@72f988bf-86f1-41af-91ab-2d7cd011db47>Yea
h, I think so. Can I start now?</v>

0:0:14.970 --> 0:0:15.450
<v 1725246c-8250-4d5b-b165-64eb54294550@72f988bf-86f1-41af-91ab-2d7cd011db47>Yea
h.</v>

"""

## My temporary starting samples
texts = [
    "Well, Evee, please put the NLP model formation as part of Amartya's work",
# direct-address item
    "Yes, I was trying to make an extensible design, but I will look for other p
ossibilities and come back to you..", # intrinsic action item
    "Can you please do it ??" ## example of action item having previous reference
s
]
```

Overall NLP workflow

This will consist of -

- Applying heuristics based on (1) POS tags (2) Dependency tree as shown below
- Thereafter, we need to check the impact %age of the heuristics and change them accordingly
    - Idea is to have a high recall i.e. more tasks are preferred and false positives are not very harmful

We also need to look for usual relationships sentences within a doc and a tokens within a sentence

In [13]:
```
import spacy
from spacy import displacy
```

In [14]:
```
# load english language model
nlp = spacy.load('en_core_web_sm',disable=['ner','textcat'])
```

In [15]:
```
# !pip install visualise-spacy-tree
```

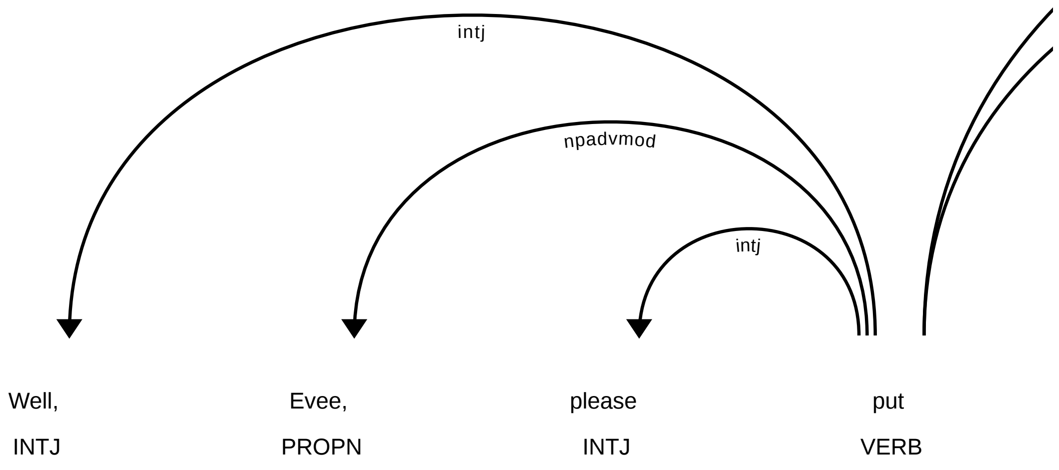
In [16]:
```
# import visualise_spacy_tree
#from IPython.display import Image, display
```

Testing out the libs with sample sentences --

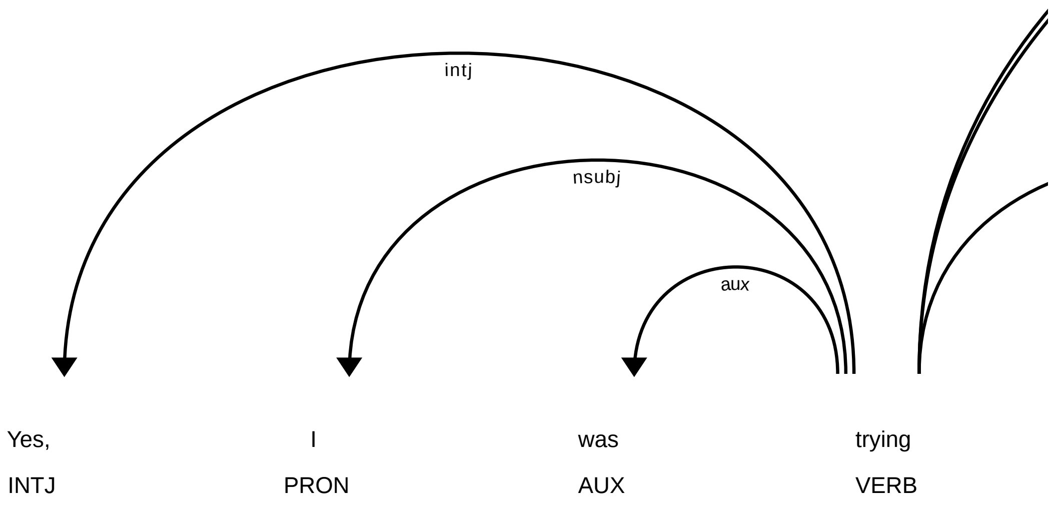
In [17]:

```
for text in texts:
    doc = nlp(text)

    for token in doc:
        print(token.text,'->',token.pos_)

    displacy.render(doc, style='dep',jupyter=True)
    # png = visualise_spacy_tree.create_png(doc)
    # display(Image(png))
```

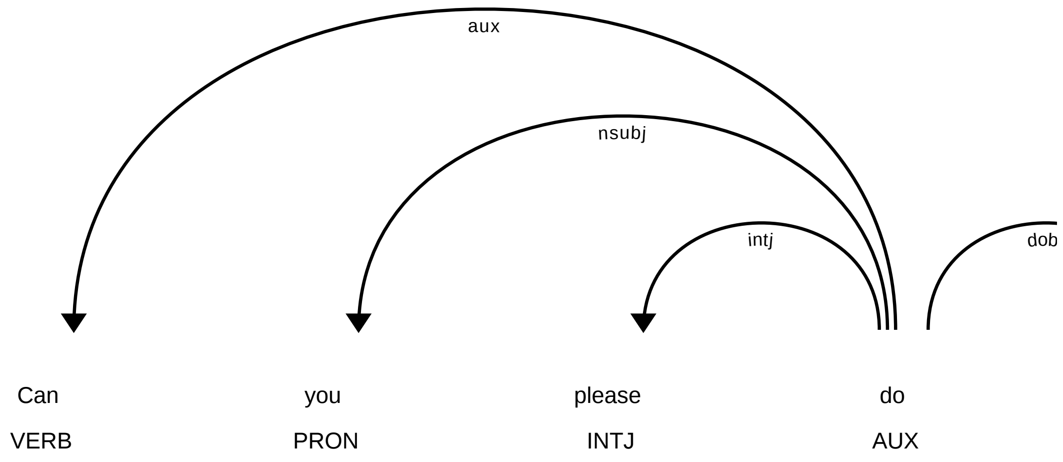
```
Well -> INTJ
, -> PUNCT
Evee -> PROPN
, -> PUNCT
please -> INTJ
put -> VERB
the -> DET
NLP -> PROPN
model -> NOUN
formation -> NOUN
as -> SCONJ
part -> NOUN
of -> ADP
Amartya -> PROPN
's -> PART
work -> NOUN
```



```
Yes -> INTJ
, -> PUNCT
I -> PRON
was -> AUX
trying -> VERB
to -> PART
make -> VERB
an -> DET
extensible -> ADJ
design -> NOUN
, -> PUNCT
but -> CCONJ
I -> PRON
will -> VERB
look -> VERB
for -> ADP
other -> ADJ
possibilities -> NOUN
and -> CCONJ
come -> VERB
```



```
Can -> VERB
you -> PRON
please -> INTJ
do -> AUX
it -> PRON
? -> PUNCT
? -> PUNCT
```


aux
nsubj
intj
dob
Can
VERB
you
PRON
please
INTJ
do
AUX