1 EXAMPLES OF INCORRECT ANNOTATIONS

in Figures 1. The *is_direct_supervision_bag* argument is annotated as *LongTensor*, and is used to send a *is_direct_supervision_bag['tokens']* call, which is converted to the *getitem* method. However, LongTesnor does not support *getitem* method[1]. ExTyper predicts it as *dict[str,Tensor]*, which supports *getitem* and passes type checking.

In the second example,the rels argument is annotated as Set[str], and is used as initializing argument of MultiLabelField, however, the MultiLabelField class annotated the formal argument as: Sequence[Union[str, int]]. Set can not be used as Sequence to initialize MultiLabelField, ExTyper predict it as List[str], which passes type check.

```
def forward(self
            is_direct_supervision_bag: torch.LongTensor,
            )
    # is all instances in this batch supervised
    is_direct_supervision_batch =
    bool(is_direct_supervision_bag['tokens'].shape[1])
```

Figure 1: Suspicious annotation: example 1

```
def text_to_instance(self,rels: Set[str]):
    fields = {"labels": MultiLabelField(rels)}
```

Figure 2: Suspicious annotation: example 2

2 EXAMPLES OF MULTIPLE CORRECT ANNOTATIONS

Second, many arguments can indeed be multiple types. For example, in Figure 3, the argument *tokens_count* used in *range* function requires a *supportindex* type[2],. Therefore, ExTyper predicts that its type can be *int*, as well as *ndarray* and *tensor*, both of which can also be used as a *supportindex* type.

```
def _positions(self, tokens_count: int):
        for i in range(tokens_count):
            ....
```

Figure 3: Argument with multiple types

3 EXAMPLES OF FALSE POSITIVES

Figure 4 presents a false positive example of Mypy, where Mypy reports type errorsno matter *cells_str* is annotated as *str* or *List[str]*, since the third line checks the compatibility of *cells_str* with the inferred type of the expression in the right of the assignment (*List[str]*).. The fourth line check the compatibility of *cells_str* with the inferred type of the expression in the right of the assignment (*str*). Neither of the annotated types can pass type checking. This is because all references of *cells_str* are analyzed as references to the argument *cells_str*, which is imprecise.

```
def cells2rle(cells_str:Union[str,List[str]])->str:
    if isinstance(cells_str, str):
        cells_str = cells_str.split("\n")
    cells_str = "\n".join(l for l in cells_str)
```

Figure 4: False positive example of Mypy

[1]https://pytorch.org/docs/stable/tensors.html
[2]https://www.python.org/dev/peps/pep-0637/