1 Problem Definition

1.1 Input

Input is a relation of OCR outputs:

```
OCR(ocrid, docid, wordid, word)
```

Each row in the relation correspond to a word output by a certain OCR, a row `ocrid, docid, wordid, word` means that the *ocrid*'th OCR believes that in document *docid*, the *wordid*'th word is *word*.

There can be structural information and visual layout information in the input listed in following relations:

```
OCR_sentence(ocrid, docid, sentid, wordid_from, wordid_to)
OCR_box(ocrid, docid, wordid, word_box)
```

1.2 Output

We want to finally populate a relation of documents predicted by our system, which contains words for each position in each document:

```
Doc(docid, wordid, word)
```

2 Procedure

2.1 Candidate generation

Generate candidate relation from inputs, based on a set of rules:

```
Candidate(docid, wordid, candid, word)
```

2.2 Feature extraction

Extract features for candidates of each word:

```
Feature(docid, wordid, candid, fname, fval)
```

2.3 Learning and Inference

2.3.1 Supervision

Our training data is

```
LabeledDoc(docid, wordid, word)
```

which is extracted from hand-labeled documents that we assume correct. We can use `LabeledDoc` to fill in respective rows in `Doc`, which are known values to our output relation:

```
Doc(docid, wordid, word)
```

We also assume that each document in `LabeledDoc` is complete, i.e. for each `docid` that appears in `LabeledDoc`, variable `word` is known for every row in `Doc`.

With the `Candidate` relation, we can transform the `Doc` relation to `DocCand`, with boolean variable `value` indicating whether each candidate is correct:

```
DocCand(docid, wordid, candid, value)
```

Therefore some values in `DocCand` relation is known, which can be used for distant supervision.

2.3.2 Inference

We use the `Feature` relation and additional inference rules to populate the relation `DocCand`, predicting boolean variable `value`.

2.3.3 Collecting results

We can populate relation `Doc` by picking the most-possible candidate for each word, with predictions in relation `DocCand` and candidate-word mappings in relation `Candidate`.

2.4 Evaluation

For evaluation, we hold out a fraction of supervision set `LabeledDoc` as `Test`, and calculate the recall of predicted words on the testing set. We always hold out **entire documents**, e.g. all rows with $docid = 1, 4, 7$ in `LabeledDoc`.

For convenience, we select `Predicted` relation for each `docid` in Test:

$$Predicted = Doc \bowtie \pi_{docid}(Test)$$

Naive evaluation:

$$Recall = \frac{|Predicted \bowtie Test|}{|Test|}$$

However the naive evaluation would wrongly punish misalignments. To fix it, we enable any no-crossing shifting that can maximize the match in the evaluation.

For each $docid = did$ in Test, denote:

$$Test_{did} = \sigma_{docid=did}(Test)$$

$$Predicted_{did} = \sigma_{docid=did}(Predicted)$$

Calculate the optimal mapping for document `did`:

$$Map_{did} = argmax_{Map_{did}}(|MappedPred \bowtie Test_{did}|)$$

$$s.t. Map_{did}(i) = j \iff \pi_{word}\sigma_{wordid=i}MappedPred = \pi_{word}\sigma_{wordid=j}Pred_{did}$$
$$\bigwedge i < j \rightarrow Map_{did}(i) < Map_{did}(j), \forall i, j \in \pi_{wordid}MappedPred$$

The optimal mapping can be calculated by dynamic programming, with a time complexity of $O(n^2)$.

We can therefore define the Precision and Recall:

$$Precision_{did} = \frac{|Map_{did}|}{|Predicted_{did}|}$$

$$Recall_{did} = \frac{|Map_{did}|}{|Test_{did}|}$$

And we use the F1 score:

$$F1_{did} = \frac{2Precision_{did}Recall_{did}}{Precision_{did} + Recall_{did}}$$

2.4.1 Examples