Phrase Extraction Algoritghm

Jakub Waszczuk

December 2018

1 Naive Algorithm

The naive phrase extraction algorithm (Alg. 1) considers all possible phrase pairs $(\bar{e}, \bar{f})$ given sentence pair $(\boldsymbol{e}, \boldsymbol{f})$ one by one. Each such phrase pair is added to the resulting set if it is consistent with the given alignment relation A. Recall that, graphically, each phrase pair $(\bar{e}, \bar{f})$ corresponds to a rectangle R in

Algorithm 1 Naive algorithm

```
F := ∅
for each i, i′: 1 ≤ i ≤ i′ ≤ n do
    for each j, j′: 1 ≤ j ≤ j′ ≤ m do
        let f̄ = (f_i, ..., f′_i)
        let ē = (e_j, ..., e′_j)
        if (f̄, ē) consistent with A then
            F := F ∪ (f̄, ē)
        end if
    end for
end for
```

the tabular representation of A, and that it is consistent if:

- R is non-empty (at least one marked cell) and
- for each column/row that intersects R, all its marked cells must be in R

The overal computation cost of the naive algorithm is $\Theta(m^2 \times n^2)$ times the cost of checking the consistency of a given rectangle (phrase pair) R w.r.t. A. The latter is not negligible, hence the total cost is unsatisfactory. There is a better way, fortunately.

2 Improved Algorithm

The improved algorithm (Alg. 2) considers all possible English phrases $\bar{e}$ for a given sentence pair. For each such English phrase, it:

1. Identifies the minimal matching foreign phrase $\bar{f}$
2. Checks if $(\bar{e}, \bar{f})$ is consistent with A

Algorithm 2 Improved algorithm

```
F := ∅
for each e_beg, e_end : 1 ≤ e_beg ≤ e_end ≤ m do
    // Find the minimal matching foreign phrase
    (f_beg, f_end) ← MinimalMatching(e_beg, e_end)
    // Extract the phrase and its possible extensions
    F ← F ∪ Extract(e_beg, e_end, f_beg, f_end)
end for
function MinimalMatching(e_beg, e_end)
    // Find the minimal matching foreign phrase
    (f_beg, f_end) ← (n + 1, 0)
    for each (e, f) ∈ A do
        if e_beg ≤ e ≤ e_end then
            f_beg ← min(f, f_beg)
            f_end ← max(f, f_end)
        end if
    end for
    return (f_beg, f_end)
end function
function Extract(e_beg, e_end, f_beg, f_end)
    // Check if at least one alignment point
    return ∅ if f_end = 0
    // Check if alignments points violate consistency
    for each (e, f) ∈ A do
        return ∅ if f_beg ≤ f ≤ f_end and (e < e_beg or e > e_end)
    end for
    E ← ∅
    f_b = f_beg
    repeat
        f_e = f_end
        repeat
            Add phrase (e_beg ... e_end, f_b ... f_e) to E
            f_e = f_e + 1
        until f_e aligned
        f_b = f_b − 1
    until f_b aligned
    return E
end function
```

3. If so, $(\bar{e}, \bar{f})$ is added to the resulting set, as well as all „extensions“ covering neighboring non-aligned words (both foreign and English)

This algorithm is faster than the naive one because, for a given $\bar{e}$, it doesn't consider all possible foreign phrases $\bar{f}$, only the relevant ones.

Proposition 1. *The improved algorithm 2 is not only faster than algorithm 1 but also correct – both calculate the same set of phrase pairs.*

Let $x_{(i,j)}$ be the phrase spanning (i, j) in sentence $\boldsymbol{x}$. To prove the above proposition we need to show that, if $(\bar{e}, f_{(f_{\text{beg}}, f_{\text{end}})})$ is consistent with A, then:

1. The minimal matching phrase $f_{(f_{\text{beg}}, f_{\text{end}})}$ is really minimal – i.e., for any foreign span (i, j) such that either $i > f_{\text{beg}}$ or $j < f_{\text{end}}$, $(\bar{e}, f_{(i,j)})$ is not consistent with A.

2. Extending $f_{(f_{\text{beg}}, f_{\text{end}})}$ with neighboring non-aligned words (both foreign and English) leads to a consistent phrase pair.

3. Extending $f_{(f_{\text{beg}}, f_{\text{end}})}$ with neighboring aligned words leads, again, to a inconsistency.

We now proceed to show the first property. Let $i > f_{\text{beg}}$ (the proof is analogous in case $j < f_{\text{end}}$). From the definition of MinimalMatch, we can see that $(e, f_{\text{beg}}) \in A$ for some $e : e_{\text{beg}} \leq e \leq e_{\text{end}}$. Since $e_{\text{beg}} \leq e \leq e_{\text{end}}$ and $(e, f_{\text{beg}}) \in A$, consistency requires that $i \leq f_{\text{beg}} \leq j$. However, that contradicts the initial assumption that $i > f_{\text{beg}}$.