Appendix

1 Cut Isnad for Individual Transmissions

Algorithm 1 Cut Isnad for Individual Transmissions

Require: isnad, edge_list

cut_isnad ← {value: isnad text for individual transmissions}
foreach edge ∈ edge_list **do**
    teacher_id, student_id ← edge[0], edge[1]
    beg_string, end_string ← "L#" + **toString**(teacher_id), "L#" + **toString**(student_id)
    transmission_text ← string match from beg_string to end_string
    cut_isnad.**append**(transmission_text)
end foreach

2 Build MoT Corpus

Algorithm 2 Build MoT Corpus

Require: cut_isnads

mot_corpus ← {value: MoT words }
mot_corpus ← initialize corpus with α MoT words `/* Where` α `is the number of known MoT words */`
need_investigation = {value: transmission text}
foreach transmission_text ∈ cut_isnads **do**
    matched_mot ← False
    foreach mot ∈ mot_corpus **do**
        if mot in transmission_text **then**
            matched_mot = True
            break();
        end if
    end foreach
    if not matched_mot **then**
        need_investigation.**append**(transmission_text);
    end if
end foreach
if len(need_investigation) == 0 **then**
    `/* manually look at` β `transmissions in need_investigation and add the undidentifed MoT words to the` α `known ones */`
    `/* repeat process until need_investigation is empty */`
end if

3 Extract MoT from Transmission Text

Algorithm 3 Extract MoT from Transmission Text

Require: mot_corpus, cut_isnads

transmission_mot = {value: **pair**(transmission, mot)}
foreach transmission ∈ cut_isnads **do**
    matched_mots ← {value: **pair**(mot, position)}
    foreach mot ∈ mot_corpus **do**
        if mot in transmission **then**
            position ← position of mot in transmission
            matched_mots.**append**((mot, position))
        end if
    end foreach
    selected_mot ← apply heuristics to select mot from matched_mots
    transmission_mot.**append**(**pair**(transmission, selected_mot))
end foreach

4 Place Transmission in Time Span

Algorithm 4 Place Transmission in Time Span

Require: teacher_bio, student_bio

```
const_lifespan = 80
const_childhood = 20
death_date_teacher ← death date from teacher_bio
death_date_student ← death date from student_bio
if birth_date ∈ teacher_bio then
    birth_date_teacher ← birth date from teacher_bio
else
    birth_date_teacher ← death_date_teacher
                         − const_lifespan
end if
if birth_date ∈ student_bio then
    birth_date_teacher ← birth date from student_bio
else
    birth_date_student ← death_date_student
                         − const_lifespan
end if
upper_bound ← min(death_date_teacher, death_date_student)
lower_bound ← max(birth_date_teacher, birth_date_student)
if upper_bound − lower_bound > const_childhood then
    lower_bound += const_childhood
end if
time_span ← (lower_bound, upper_bound)
```