Which-hunting in medieval England

Supplementary materials

Robert Truswell and Nikolas Gisborne (University of Edinburgh)

Introduction

These materials are intended for the reproduction of figures and main quantitative results presented in our *Canadian Journal of Linguistics* paper '*Which*-hunting in medieval England'. We want to make it clear that these figures are **not** the main analysis presented in that paper, but just visualizations of some patterns in the history of English relative clauses that support our main claims, that *which*-relatives and *what*-relatives behave differently, and *which*-relatives with and without an NP restrictor behave differently. The purpose of making these scripts publicly available is to encourage further exploration of these patterns, and to allow scrutiny of some of our methods and modelling assumptions.

Files included

- This guide:
  - TruswellGisborne_supplementary_materials_notes.pdf
- Six CorpusSearch coding query files:
  - LeftPeripheries_YCOE.c
  - LeftPeripheries_PPCHE.c
  - LeftPeripheries_PLAEME.c
  - Nonref_antecedent.c
  - Nonref_car.c
  - Nonref_extraposition.c
- A CorpusSearch definitions file referred to by the coding queries:
  - WhRel.def
- One CorpusSearch script for stripping codes output by coding queries:
  - OoosIds.q
- One manually edited set of codes:
  - Nonref_car_manual.cod.ooo
- Four files containing textual metadata:
  - YCOE_match_IDs_to_files.csv
  - Penn_match_IDs_to_files.csv
  - Allfiles_basic_info.csv
  - PLAEME_basic_info.csv
- An R script for generating the figures and estimating frequency of the key *no NP which NP* pattern:
  - CJL.R

Not included but essential

To reproduce our results, you will also need access to:

- Four corpora:
  - The York–Toronto–Helsinki Parsed Corpus of Old English Prose (YCOE).
  - The Penn Parsed Corpora of Historical English (PPCHE).
  - The Parsed Corpus of Middle English Poetry (PCMEP).
  - A Parsed Linguistic Atlas of Early Middle English (PLAEME).
- CorpusSearch, for running corpus queries.
- R, plus the relevant packages.

All of these corpora except YCOE are still actively maintained and developed. Moreover, we use a local version of all corpora with parsing errors corrected opportunistically as they are identified. Running the scripts on 'fresh' (recently downloaded, unmodified) copies of the corpora will produce slightly different results to those which appear in the paper, but we would not expect the differences to be significant.

What the files do

LeftPeripheries….c

These three coding queries code all CPs for three different pieces of information:

- Column 1: Is the CP a headed relative (`Rel`), a clause-adjoined relative (`Car`), a free relative (`Frl`), or none of the above (`X`)?
- Column 2: Is the relativizer a form of *which* (`Which`), a form of *what* (`What`), another *wh*-form (`HW`), or none of the above (`X`)?
- Column 3: Does the relativizer have an NP complement (`N`) or not (`X`)?

Nonref….c

These three coding queries code relatives for two different pieces of information:

- Column 1: Is the antecedent headed by a form of *no* (`No`), a form of *few* (`Few`), a form of *each* (`Each`), a form of *every* (`Every`), a form of *little* (`Little`), or something else (`X`)?
- Column 2: Is the relative headed by *which* with NP complement (`WhichN`), bare *which* (`Which`), is the relative extraposed (`Extraposed`), is it some other type of relative (`Rel`), or is there no relative after all (`X`)?

Finding the antecedent of a relative is straightforward if the relative is NP-internal, slightly less straightforward if extraposed, and not explicitly annotated in Penn-format corpora if treated as clause-adjoined (annotation guidelines in fact mandate the latter case for all headed relatives with NP complements). Our strategy was to treat these three cases with three separate queries:

- `Nonref_antecedent.c` : looks for NP-internal relatives.
- `Nonref_extraposed.c` : looks for extraposed relatives.
- `Nonref_car.c` : looks for clause-adjoined relatives.

The latter case required further manual annotation of the antecedent—these annotations can be found in `Nonref_car_manual.cod.ooo`

These three files are all designed to run only on PPCHE and PCMEP. The other two corpora concern earlier periods, in which there are close to 0 headed *which*-relatives, so questions of the antecedent of *which*-relatives simply do not arise.

WhRel.def

This lists orthographic variants for all of the words referred to in the coding queries, e.g. *which*, *no*. These lists were compiled by exhaustive search and classification of forms with the relevant tags. They are not 100% accurate because of potential orthographic overlap between different words, but the amount of noise is in practice very small.

Workflow

1. Use CorpusSearch to run all coding queries on relevant corpus files. Exactly how you do this will depend on your local configuration, but, for instance, to run LeftPeripheries_YCOE.c, you should type approximately: `java -cp PATH_TO_CORPUSSEARCH/ CS_2.003.04.jar LeftPeripheries_YCOE.c PATH_TO_YCOE/psd/*psd` PPCHE and PCMEP have the same annotation format in all relevant respects. YCOE and PLAEME are different (YCOE uses different tags, PLAEME has lemmas), so different coding queries are designed to be run on different corpora (this is easiest to achieve if you create a single directory containing all PPCHE and PCMEP files—the filenames unambiguously indicate which file comes from which corpus). The output files will have identical names to the coding queries, except with extension `.cod` rather than `.c`.

    - Run LeftPeripheries_YCOE.c on YCOE.
    - Run LeftPeripheries_PLAEME.c on PLAEME.
    - Run all other coding queries on PPCHE and PCMEP simultaneously.

2. Use CorpusSearch to run `OoosIds.q` on each of the `.cod` files just generated. For example: `java -cp PATH_TO_CORPUSSEARCH/CS_2.003.04.jar OoosIds.q LeftPeripheries_YCOE.cod` The output of this is a file with the extension `.cod.ooo`, containing just the codes generated by the coding queries, with the ID of the sentence token appended.
3. Manually edit all six `.cod.ooo` files: replace `@` with `:` (this allows the files to be treated as functionally equivalent to `.csv` files, with separator `:`). Also remove the part of the ID tag which uniquely identifies the sentence token rather than the text. For most `.cod.ooo` files, this involves removing the material after `,`, but because of the different format of YCOE, in that case you must remove the material after the final `:`. This can be done using `sed`, or search-and-replace facilities in your favourite text editor.
4. Because of the PPCHE convention of treating clause-adjoined relatives as separate sentence tokens in many cases, we had to manually edit `Nonref_car.cod` to add information about the antecedent of relevant relatives tagged as clause-adjoined relatives (tags starting with `CP-CAR`). To inspect our manual annotations, you can compare the file `Nonref_car_manual.cod.ooo` (included) with the file `Nonref_car.cod.ooo` automatically generated by the above process. The R script provided only makes reference to `Nonref_car_manual.cod.ooo`.
5. Run `CJL.R`. This script calls all the `.cod.ooo` files generated, as well as the metadata `.csv` files provided.