Wh-Questions Tgrep2 Corpus Annotation Guide

Morgan Moyer Version: November 5, 2020

---

1 Macros

1. **Disfluencies @ DISFL:**
   /EDITED|UH|PRN|-UNF/

2. **Wh-Node @ WH:**
   /WP|WRB|WDT/ (The word-level wh-node)
   !< /ˆt|ˆT/ (that doesn't termininate in a that/That)
   !≫ @DISFL (isn't dominated by a disfluency)
   ≫ /ˆS/ (but is dominated by an S node)
   !> /ˆN|ˆA/ (isn't parented by any NPs or ADV/ADJPs)
   ≫ /ˆWH/

2 Categorical Variables

1. **WhPhraseType**:
   Returns "complex" if @WH has a sister (e.g., which children); returns "monomorphemic" otherwise (e.g., who).

   (a) Complex:
   @WH [$ /ˆN|ˆD|ˆJ|ˆRB|ˆRP|ˆPP/ | ≫ (/ˆWH/ $ /ˆAD|ˆJJ|ˆNP|ˆPP/)]

   (b) Monomorphemic:
   @WH [!$ /ˆN|ˆD|ˆJ|ˆRB|ˆRP|ˆPP/ | !≫ (/ˆWH/ $ /ˆAD|ˆJJ|ˆNP|ˆPP/)]

2. **ModalPresent**:
   Returns "yes" if there is a modal auxiliary (can, could, shall, should, may, might, must) in the clause that is sister to the WH-phrase node that dominates the @WH (where we can find coffee); Returns "no" if there isn't any.

   (a) Yes:
   @WH ≫ (/ˆWH/ ≫ /ˆS/ $ (/S|SQ/ ≪ /MD/))

   (b) No:
   @WH !≫ (/ˆWH/ ≫ /ˆS/ $ (/S|SQ/ ≪ /MD/))

3. **HaveNeed**:
   Returns 'yes' if there's a have to or need to construction. Returns 'no' otherwise.

   (a) Yes:
   @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVP/ ≪ /need|needs|needed|have|has|had|'ve|'d/) ≪ /TO/))

(b) No:
@WH !≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVP/ ≪ /need|needs|needed|have|has|had|'ve|'d/) ≪ /TO/))

4. **Finite**
Returns "finite" if the question contains a tensed (=non-modal) verb (where they found coffee); returns "infinite" if the question contains an infinitival clause (where to find coffee)

(a) **Finite**:
@WH ≫ (/ˆWH/ !$ (/ˆS/ !$ /ˆVB/ < (/ˆNP/ < /-NONE-/ $. (/ˆVP/ < /TO/))))

(b) **Infinitival**:
@WH ≫ (/ˆWH/ $ (/ˆS/ !$ /ˆVB/ < (/ˆNP/ < /-NONE-/ $. (/ˆVP/ < /TO/))))

5. **QuestionType**
Categorizes the kind of wh-question. We are mostly interested in Root (Where can we find coffee?) and Embedded (Dana knows where we can find coffee), and possibly Embedded Adjuncts. Embedded Adjuncts may look on the surface like Embedded Clauses, the difference only being whether the wh-clause is a complement to a verb or an adjunct.

CRITICAL QUESTIONTYPES

(a) **Root**
Wh-clauses that have interrogative illocutionary force (e.g., Where can I find coffee?). Are not complements to adjectives/verbs or children of Ns
@WH
≫ (/ˆSQ|ˆSBARQ/ [!$ /ˆVB|ˆJJ|ˆBE|ˆRB/ | !> /ˆN/])
!≫ /SBAR-NOM/

(b) **Embedded**
Wh-clauses that are complements (sisters) to a verb (VB) or an adjective (JJ), OR that are embedded in PP which are complements of verbs or adjectives.
@WH ≫ (/ˆSBAR/ [$ /ˆVB|ˆJJ|ˆBE|ˆRB/] | [> /ˆPP/ $ /ˆJJ|ˆVB|ˆBE|ˆRB/ ,, /ˆV/]
!> /ˆNP/ !$ /ˆNN/!= /SBAR-TMP|SBAR-LOC/)

NON-CRITICAL QUESTION-TYPES
These are tags meant to weed out spurious root/embedded questions.

(a) **Relative**
The Wh-clause modifies a noun. (Dana saw the boy who ate Captain Crunsh).
@WH
≫ (/ˆS/ [> /ˆN/ | > (/ˆP/ > /ˆN/)])
!≫ (/ˆSBAR/ [$ /ˆVB|ˆJJ|ˆBE|ˆRB/] | [> (/ˆPP/ $ /ˆVB|ˆJJ|ˆBE|ˆRB/)])

(b) **EmbAdjunct**
Wh-clauses that are not complements to a verb/adjective, but linearly follow the verb/adjective (i.e., that look like embedded clauses from a linear perspective).
@WH
≫ (/ˆSBAR/ $ /ˆVP|ˆAD/ !$ /ˆVB|ˆJJ|ˆBE|ˆRB/
[!> (/ˆS/ > /TOP/) | !> /TOP/] | = /SBAR-TMP—SBAR-LOC/)

(c) **Adjunct**
Wh-clauses that are neither complements of verbs/adjectives, nor the children of PPs

that are complements of verbs/adjectives @WH
[≫ (/ˆSBAR/ [> (/ˆPP/ !$ /ˆVB|ˆJJ|ˆBE|ˆRB/)] | [!$ /ˆVB|ˆJJ|ˆBE|ˆRB/]) | ≫ /SBAR-LOC—SBAR-TMP/]

(d) **Subject**
Wh-clauses that are initial.
@WH
[≫ /SBAR-SBJ/ | ≫ (/ˆS/=s1 ≫ /TOP/ [!≫ (/ˆV/ ≫ =t1) | !,, (/ˆN/ ≫ =t1)])]

(e) **Fragment**
Wh-clauses that are fragments.
@WH
≫ (/ˆSBAR/ > /FRAG/)

(f) **Exclam**
Wh-clauses that are exclamatives (don't have a verb), e.g., what a meal!
@WH
≫ (/ˆS/ !≪ /ˆV/)
!≫ /ˆSBAR-NOM/

6. **DegreeQ**
Returns 'yes' if the wh-clause is a degree-question (how many cups of salt do we need?); returns 'no' otherwise.

   (a) Yes: @WH $ /ˆJJ|ˆRB/

   (b) No: @WH !$ /ˆJJ|ˆRB/

7. **WhAll**
Returns 'yes' if the wh-phrase contains an all; returns 'no' otherwise.

   (a) Yes: @WH ≫ (/ˆWH/ ≪ (/ˆNP/ < (/ˆDT/ < /all/) !< /ˆN/))

8. **SubjectAux**
Returns 'yes' if the wh-clause contains subject-aux-inversion.

   (a) Yes: @WH ≫ (/ˆSBAR/ ≪ /ˆSQ/)

9. **Determiner Subject**

   (a) Yes:
   @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP-SBJ/ ≪ /DT/)))

   (b) No:
   @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP-SBJ/ !≪ /DT/)))

10. **DeterminerNonSubject**:

    (a) Yes:
    @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP/ ≪ /DT/ ≫ /ˆVP/)))

    (b) No:
    @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP/ !≪ /DT/ ≫ /ˆVP/)))

11. **IdentityQ**:

(a) YEs:
@WH ≫ (/ˆWH/ $ (/ˆS/=s1 ≪ (/'s|'m|'re|am|is|are|was|were/=vb1 !.. (/ˆVB|ˆJJ|ˆRB/ ≫ =s1 !< =vb1) !.. /ˆPP/)))

(b) No:
@WH !≫ (/ˆWH/ $ (/ˆS/=s1 ≪ (/'s|'m|'re|am|is|are|was|were/=vb1 !.. (/ˆVB|ˆJJ|ˆRB/ ≫ =s1 !< =vb1) !.. /ˆPP/)))

(c) **MatrixNegPresent**

i. Yes:
@WH ≫ (/ˆWH/ ¿ (/ˆSBAR/ [$ /ˆVB|ˆBE|ˆJJ|ˆRB//=vb |
> (/ˆPP/=pp $ /ˆVB|ˆBE|ˆJJ|ˆRB/)] !> /ˆNP/ !$ /ˆNN/ ≫ (/ˆVP/ ≪ /not|n't/ [≪ =vb | ≪ =pp])))

ii. No:
@WH !≫ (/ˆWH/ ¿ (/ˆSBAR/ [$ /ˆVB|ˆBE|ˆJJ|ˆRB//=vb |
> (/ˆPP/=pp $ /ˆVB|ˆBE|ˆJJ|ˆRB/)] !> /ˆNP/ !$ /ˆNN/ ≫ (/ˆVP/ ≪ /not|n't/ [≪ =vb | ≪ =pp])))

(d) **EmbeddedNegPresent**

i. Yes:
@WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVB|ˆBE/ !, /ˆVB|ˆBE/ $ (/ˆRB/ < /not|n't/))))

ii. No:
@WH !≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVB|ˆBE/ !, /ˆVB|ˆBE/ $ (/ˆRB/ < /not|n't/))))

3 String Variables

1. **MatrixNegation**: Print matrix negation.
@WH ≫ (/ˆWH/ > (/ˆSBAR/ [$ /ˆVB|ˆBE|ˆJJ|ˆRB//=vb |
> (/ˆPP/=pp $ /ˆVB|ˆBE|ˆJJ|ˆRB/)] !> /ˆNP/ !$ /ˆNN/ ≫ (/ˆVP/ ≪ /not|n't/=print [≪ =vb | ≪ =pp])))

2. **MatrixPred1**: Print all the matrix predicates who have the wh-clause as complement.
@WH ≫ (/ˆWH/ ≫ /ˆS/=s1 ≫ (/ˆS/=s2 $ (/ˆVB|ˆBE|ˆJJ|ˆRB/=print !,, (/ˆVB|ˆBE/ ≫ =s1 $ /ˆNP-SBJ/) ,, (/ˆNP-SBJ/ ≫ =s1)) ≫ =s1)) FIX?

3. **MatrixPred2**: Print all the second matrix predicates.
@WH ≫ (/ˆWH/ ≫ /ˆS/=s1 ≫ (/ˆS/=s2 $ (/ˆVB|ˆBE|ˆJJ|ˆRB/=print ,, (/ˆVB|ˆBE/ ≫ =s1 $ /ˆNP-SBJ/=np1) ,, (=np1 ≫ =s1)) ≫ =s1))

4. **MatVerbPart**: Print the matrix predicates who have wh-clause as complement to PP (e.g., surprised by, agree on, depend on).
@WH ≫ (/ˆWH/ ≫ /ˆS/=s1 ≫ (/ˆS/=s2 > (/ˆPP/ $ /ˆVB|ˆBE|ˆJJ|ˆRB/=print)))

5. **Wh**: Print the wh-word terminal node.
@WH < /ˆw|ˆh/=print

6. **Modal**: Print any modals following the @WH node.
@WH ≫ (/ˆWH/ ≫ /ˆS/ $ (/S|SQ/ ≪ /MD/=print))

7. **HaveNeed**: Print any have to / need to modal constructions @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVP/ ≪ /need|needs|needed|have|has|had|'ve|'d/=print) ≪ /TO/))

8. **EmbeddedNegation**: Print negation in the wh-clause.
   @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆVB|ˆBE/ !, /ˆVB|ˆBE/ $ (/ˆRB/ < /not|n't/=print))))

9. **Verb1**: Print the first verb in the wh-clause.
   @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/|ˆVB|ˆBE/=print !, /ˆVB|ˆBE/)))

10. **Verb2**: Print the second verb/predicate in the wh-clause.
    @WH ≫ (/ˆWH/ $ (/ˆS/=s1 ≪ (/ˆVB|ˆJJ|ˆBE/=print ,, (/ˆVB|ˆBE/ ≫ =s1))))

11. **Verb3**: Print the third verb/predicate in the wh-clause.
    @WH ≫ (/ˆWH/ $ (/ˆS/=s1 ≪ (/ˆVB|ˆJJ|ˆBE/=print ,, (/ˆVB|ˆJJ|ˆBE/ ,, (/ˆVB|ˆBE/ ≫ =s1)))))

12. **DeterminerSubject**: Print any Subj-NP with determiners.
    @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP-SBJ/=print ≪ /DT/))

13. **DeterminerNonSubject**: Print any NPs in the predicate with determiners.
    @WH ≫ (/ˆWH/ $ (/ˆS/ ≪ (/ˆNP/=print ≪ /DT/ ≫ /ˆVP/))

14. **FullWhPhrase**: Print the full WH-Phrase.
    @WH ≫ (/ˆWH/=print $ /S|SQ/ ≫ /ˆS/ ≫ /ˆN|ˆJ|ˆRB|ˆPP|ˆA/)

15. **WhParse**: Print the Wh-phrase with POS. /WP—WRB—WDT/=print !< /ˆt—ˆT/

16. **Sentence**: Print the full sentence in which the @WH node appears.
    @WH ≫ (∗=print !> ∗)

17. **Question**: Print the full wh-clause.
    @WH ≫ /ˆS/=print

18. **QuantifiedPredicate**: Print any quantifiers in the wh-clause predicate.
    @WH
    ≫ (/ˆWH/ > (/ˆS/ ≪ (/ˆVP/
    ≪ /all|every|some|one|any|most|least|more|most|much/=print)))

19. **QuantifiedSubject**: Print any quantifiers in the wh-clause subject.
    @WH
    ≫ (/ˆWH/ > (/ˆS/ ≪ (/ˆNP-SBJ/
    ≪ /all|every|some|one|any|most|least|more|most|much/=print)))