**XQuery Examples Resource**

***Queries in XQuery for the* Inaugural Rhetorical Corpus *and the* Inaugural Training Corpus**

Table of Contents (press CTRL + left mouse button to jump to query)

[Selecting nodes 4](#_Toc411359862)

[Selects all anaphora within their tags: 4](#_Toc411359863)

[Selects all singular nouns within their tags: 4](#_Toc411359864)

[Selects all nouns irrespective of number within their tags: 4](#_Toc411359865)

[Selects text of first colons in tricola: 4](#_Toc411359866)

[Selects all sentences that do not host any <isoco> element: 4](#_Toc411359867)

[Selects all sentences that do not host either<isoco> or <anaph> elements: 4](#_Toc411359868)

[Returns the 'c5' values in each sentence: 4](#_Toc411359869)

[Selects all words (<w> elements) in those sentences that do not host either<isoco> or <anaph> elements: 4](#_Toc411359870)

[Selects all sentences with less than (<) 5 words (excluding from the count instances of genitive –*s* tagged as a word) 5](#_Toc411359871)

[Selects those sentences that host exactly 8 <isoco> elements: 5](#_Toc411359872)

[Using the count() function 5](#_Toc411359873)

[Counts number of sentences: 5](#_Toc411359874)

[Counts number of sentences per paragraph (<p>) element: 5](#_Toc411359875)

[Counts number of words: 5](#_Toc411359876)

[Counts words per paragraph (excluding instances of genitive –*s* tagged as <w>): 5](#_Toc411359877)

[Counts genitive-*s* tagged as a word (e.g., <w id="15.20" c5="POS">'s</w>): 6](#_Toc411359878)

[Counts number of anaphora: 6](#_Toc411359879)

[Counts number of isocola: 6](#_Toc411359880)

[Counts number of words in all anaphora: 6](#_Toc411359881)

[Counts number of words in all isocola: 6](#_Toc411359882)

[Counts number of words for each component of anaphora: 6](#_Toc411359883)

[Counts number of <isoco> elements per sentence: 6](#_Toc411359884)

[Counts number of <isoco> and <anaph> elements per sentence: 7](#_Toc411359885)

[Counts number of words per colon for dicola: 7](#_Toc411359886)

[Counts number of words per colon for tricola: 7](#_Toc411359887)

[Counts number of words per colon for tetracola: 7](#_Toc411359888)

[Traversing axes 8](#_Toc411359889)

[Counts the number of words in all anaphora that are children to <isoco> elements: 8](#_Toc411359890)

[Selects all isocola that host an <isoco> child or <isoco> descendant: 8](#_Toc411359891)

[Selects all isocola whose ancestor is an <isoco> element: 8](#_Toc411359892)

[Selects all anaphora which are contained within (i.e. descendants to) isocola: 8](#_Toc411359893)

[Selects all anaphora which are *not* contained within (i.e. descendants to) isocola: 8](#_Toc411359894)

[Selects those possessive determiners that are immediately followed by a noun (singular or plural): 8](#_Toc411359895)

[Selects those nouns (irrespective of number) that are immediately preceded by a possessive determiner: 9](#_Toc411359896)

[Returns those singular nouns immediately following a possessive determiner: 9](#_Toc411359897)

[Using the position() and last() functions 10](#_Toc411359898)

[Returns the first word at every axis: 10](#_Toc411359899)

[Returns the last word at every axis: 10](#_Toc411359900)

[Returns the last word in every last <isoco> element in the sentence: 10](#_Toc411359901)

[Returns the last word in each sentence: 10](#_Toc411359902)

[Returns all nouns (irrespective of number) that occur in a window of 1-4 words after a possessive determiner: 10](#_Toc411359903)

[Advanced queries 11](#_Toc411359904)

[Computing collocates 11](#_Toc411359905)

[Returns all words occurring after the word “our” in a 2-word window on the right 11](#_Toc411359906)

[Returns all nouns (irrespective of number) that occur in a window of 1 to 3 words after a possessive determiner: 11](#_Toc411359907)

[Returns all verb forms (regardless of verb type) occurring in a window of 1 to 3 words after a *wh*-word (in CLAWS5 *wh*-words can be tagged as 'PNQ' (e.g., *who*), 'AVQ' (e.g., *why*, *when*), and 'DTQ' (e.g., *what*, *which*, *whose*): 11](#_Toc411359908)

[Returns all adjectives occurring in a window of 1 to 3 words before a noun (regardless of number and type) occurring in a window of 1 to 3 words after a *wh*-word (in CLAWS5 *wh*-words can be tagged as 'PNQ' (e.g., *who*), 'AVQ' (e.g., *why*, *when*), and 'DTQ' (e.g., *what*, *which*, *whose*): 12](#_Toc411359909)

[Returns all verb forms (of any type) occurring in a window of 1 to 3 words before an adverb particle (e.g., *up*, *out*): 12](#_Toc411359910)

[Returns all words occurring in a window of 1 to 2 words left and right of a verb form (of any type): 12](#_Toc411359911)

[Returns all adjectives occurring in a window of 1 to 2 words left and right of a noun (of any number and type): 12](#_Toc411359912)

[Computing n-grams 13](#_Toc411359913)

[Produces an ordered list of all 3-grams: 13](#_Toc411359914)

[Returns all 4-POS-grams: 13](#_Toc411359915)

[Constructing frequency lists 14](#_Toc411359916)

[Constructs a list of word types in descending order of frequency: 14](#_Toc411359917)

[Constructs a list of POS tags in ascending order of frequency: 14](#_Toc411359918)

[Constructs a list of word token bigrams in descending order of frequency: 14](#_Toc411359919)

[Constructs a list of POS trigrams in descending order of frequency: 15](#_Toc411359920)

[Calculating positions 17](#_Toc411359921)

[Calculates positions of possessive pronouns (DPS): 17](#_Toc411359922)

[Calculates positions of anaphora; also returns corresponding values on 'repeat' attributes: 17](#_Toc411359923)

[Calculates positions of first colons in tricolons: 17](#_Toc411359924)

[Calculates positions of last colons: 17](#_Toc411359925)

[Calculates positions of the last word in the last colon; also returns that last word and its 'c5' value: 18](#_Toc411359926)

[Constructing Key Word In Context (KWIC) displays 19](#_Toc411359927)

[Constructs a KWIC display for the search word “nation” in a 5-word window left and right of the node (L5-R5) 19](#_Toc411359928)

[Constructs a L3-R3 KWIC output for all adjectives (tagged 'AJ0') and all surrounding POS values 19](#_Toc411359929)

# Selecting nodes

## Selects all anaphora within their tags:

for $anaphora := //anaph

return

$anaphora

## Selects all singular nouns within their tags:

for $nouns in //w[@c5='NN1']

return

$nouns

## Selects all nouns irrespective of number within their tags:

for $nouns in //w[starts-with(@c5,'NN')]

return

$nouns

## Selects text of first colons in tricola:

for $tricola in //isoco[@ctype='tri']

return

string-join($tricola[@sequ='first']//w/text(),' ')

Note: instead of text(), string() can be used

## Selects all sentences that do not host any <isoco> element:

for $sentences in //s[not(descendant::isoco)]

return

$sentences

Note: negative restriction is achieved by not() function

## Selects all sentences that do not host either<isoco> or <anaph> elements:

for $sentences in //s[not(descendant::isoco) and not(descendant::anaph)]

return

$sentences

## Returns the 'c5' values in each sentence:

for $s in //s

return

string-join($s//w[@c5]/string(@c5),' ')

## Selects all words (<w> elements) in those sentences that do not host either<isoco> or <anaph> elements:

for $sentences in //s[not(descendant::isoco) and not(descendant::anaph)]

let $words := $sentences//w[not(c5='PUN')]

return

$words

## Selects all sentences with less than (<) 5 words (excluding from the count instances of genitive –*s* tagged as a word)

for $sentences in //s[count(descendant::w[not(c5='PUN')]) < 5]

return

$sentences

## Selects those sentences that host exactly 8 <isoco> elements:

for $sentences in //s[count(descendant::isoco) = 8]

return

$sentences

# Using the count() function

## Counts number of sentences:

for $whole\_doc in //wtext

let $n\_sentences := count($whole\_doc//s)

return

$n\_sentences

## Counts number of sentences per paragraph (<p>) element:

for $paragraphs in //p

return

count($paragraphs//s)

## Counts number of words:

for $whole\_doc in //wtext

let $n\_words := count($whole\_doc//w)

return

$n\_words

## Counts words per paragraph (excluding instances of genitive –*s* tagged as <w>):

for $paragraphs in //p

return

count($paragraphs//w[not(@c5="PUN")

## Counts genitive-*s* tagged as a word (e.g., <w id="15.20" c5="POS">'s</w>):

for $words in //wtext

let $s := count($words//w[@c5="POS"])

return

$s

## Counts number of anaphora:

for $whole\_doc in //wtext

let $n\_anaphora := count($whole\_doc//anaph)

return

$n\_anaphora

## Counts number of isocola:

for $whole\_doc in //wtext

let $n\_isocola := count($whole\_doc//isoco)

return

$n\_isocola

## Counts number of words in all anaphora:

for $whole\_doc in //wtext

let $n\_words\_anaphora := count($whole\_doc//anaph//w)

return

$n\_words\_anaphora

## Counts number of words in all isocola:

for $whole\_doc in //wtext

let $n\_words\_isocola := count($whole\_doc//isoco//w)

return

$n\_words\_isocola

## Counts number of words for each component of anaphora:

for $anaphora in //anaph

let $w := count($anaphora//w)

return

$w

Note: no restrictions to the context node are necessary because <anaph>elements, in this corpus, do not nest in one another.

## Counts number of <isoco> elements per sentence:

for $sentences in //s

let $n\_isocola := count($sentences//isoco)

return

$n\_isocola

## Counts number of <isoco> and <anaph> elements per sentence:

for $sentences in //s

let $n\_isocola := count($sentences//isoco)

let $n\_anaphora := count($sentences//anaph)

return

concat($n\_isocola,';', $n\_anaphora)

## Counts number of words per colon for dicola:

for $isocola in //isoco[@ctype='di']

let $words\_first.colon := count($isocola[@sequ='first']//w[not(@c5="POS")])

let $words\_second.colon := count($isocola[@sequ='second']//w[not(@c5="POS")])

let $sequ\_value := $isocola[@sequ]

return

concat($isocola[@sequ]/string(@sequ),';',$words\_first.colon,';',$words\_second.colon)

## Counts number of words per colon for tricola:

for $isocola in //isoco[@ctype='tri']

let $words\_first.colon := count($isocola[@sequ='first']//w[not(@c5="POS")])

let $words\_second.colon := count($isocola[@sequ='second']//w[not(@c5="POS")])

let $words\_third.colon := count($isocola[@sequ='third']//w[not(@c5="POS")])

let $sequ\_value := $isocola[@sequ]

return

concat($isocola[@sequ]/string(@sequ),';',$words\_first.colon,';',$words\_second.colon,';',$words\_third.colon)

## Counts number of words per colon for tetracola:

for $isocola in //isoco[@ctype='tet']

let $words\_first.colon := count($isocola[@sequ='first']//w[not(@c5="POS")])

let $words\_second.colon := count($isocola[@sequ='second']//w[not(@c5="POS")])

let $words\_third.colon := count($isocola[@sequ='third']//w[not(@c5="POS")])

let $words\_fourth.colon := count($isocola[@sequ='fourth']//w[not(@c5="POS")])

let $sequ\_value := $isocola[@sequ]

return

concat($isocola[@sequ]/string(@sequ),';',$words\_first.colon,';',$words\_second.colon,';',$words\_third.colon,';',$words\_fourth.colon)

# Traversing axes

## Counts the number of words in all anaphora that are children to <isoco> elements:

for $whole\_text in //wtext

let $n\_words := count($whole\_text//anaph[parent::isoco]//w[not(@c5="POS")])

return

$n\_words

## Selects all isocola that host an <isoco> child or <isoco> descendant:

for $isocola in //isoco

let $isoco\_descendants := $isocola[child::isoco]

return

$isoco\_descendants

## Selects all isocola whose ancestor is an <isoco> element:

for $isocola in //isoco

let $isoco\_descendants := $isocola[ancestor::isoco]

return

$isoco\_descendants

## Selects all anaphora which are contained within (i.e. descendants to) isocola:

for $anaphora in //anaph

let $anaph\_in\_isoco := $anaphora[ancestor::isoco]

return

$anaph\_in\_isoco

## Selects all anaphora which are *not* contained within (i.e. descendants to) isocola:

for $anaphora in //anaph

let $anaph\_out\_isoco := $anaphora[not(ancestor::isoco)]

return

$anaph\_out\_isoco

## Selects those possessive determiners that are immediately followed by a noun (singular or plural):

for $dps in //w[@c5='DPS' and following::w[1][starts-with(@c5,'NN')]]

return

$dps

## Selects those nouns (irrespective of number) that are immediately preceded by a possessive determiner:

for $nn in //w[starts-with(@c5,'NN') and preceding::w[1][@c5='DPS']]

return

$nn

## Returns those singular nouns immediately following a possessive determiner:

for $dps in //w[@c5='DPS']

let $nn := $dps/following-sibling::w[1][@c5='NN1']

return

$nn

# Using the position() and last() functions

## Returns the first word at every axis:

for $first\_word in //w[position()=1]

return

$first\_word

## Returns the last word at every axis:

for $last\_word in //w[last()]

return

$last\_word

## Returns the last word in every last <isoco> element in the sentence:

for $s in //s

let $last\_word := $s//descendant::isoco[last()]//w[last()]

return

$last\_word

## Returns the last word in each sentence:

for $s in //s

let $last\_word\_sentence := $s/descendant::w[last()]

return

$last\_word\_sentence/string()

## Returns all nouns (irrespective of number) that occur in a window of 1-4 words after a possessive determiner:

for $dps in //w[@c5='DPS']

let $nn := $dps/following-sibling::w[position()=1 to 4][starts-with(@c5,'NN')]

return

$nn

# Advanced queries

# Computing collocates

## Returns all words occurring after the word “our” in a 2-word window on the right

for $our in //w[text()='our']

let $w\_slot1 := $our/following::w[1]

let $w\_slot2 := $our/following::w[2]

return

concat($our[@id]/string(@id),'; ',$our/string(),' ',$w\_slot1/string(),' ',$w\_slot2/string())

Note: using text() it is possible to search for text independently of its POS value

## Returns all nouns (irrespective of number) that occur in a window of 1 to 3 words after a possessive determiner:

for $dps in //w[@c5='DPS']

let $nn\_pos1 := $dps/following::w[position()=1][starts-with(@c5,'NN')]

let $nn\_pos2 := $dps/following::w[position()=2][starts-with(@c5,'NN')]

let $nn\_pos3 := $dps/following::w[position()=3][starts-with(@c5,'NN')]

return

concat($dps[@id]/string(@id),';',$dps/string(),';',$nn\_pos1/string(),';',$nn\_pos2/string(),';',$nn\_pos3/string())

## Returns all verb forms (regardless of verb type) occurring in a window of 1 to 3 words after a *wh*-word (in CLAWS5 *wh*-words can be tagged as 'PNQ' (e.g., *who*), 'AVQ' (e.g., *why*, *when*), and 'DTQ' (e.g., *what*, *which*, *whose*):

for $wh in //w[ends-with(@c5,'Q')]

let $v\_pos1 := $wh/following::w[position()=1][starts-with(@c5,'V')]

let $v\_pos2 := $wh/following::w[position()=2][starts-with(@c5,'V')]

let $v\_pos3 := $wh/following::w[position()=3][starts-with(@c5,'V')]

return

concat($wh[@id]/string(@id),';',$wh/string(),';',$v\_pos1/string(),';',$v\_pos2/string(),';',$v\_pos3/string())

## Returns all adjectives occurring in a window of 1 to 3 words before a noun (regardless of number and type) occurring in a window of 1 to 3 words after a *wh*-word (in CLAWS5 *wh*-words can be tagged as 'PNQ' (e.g., *who*), 'AVQ' (e.g., *why*, *when*), and 'DTQ' (e.g., *what*, *which*, *whose*):

for $n in //w[starts-with(@c5,'N')]

let $adj\_pos-1 := $n/preceding::w[position()=1][@c5='AJ0']

let $adj\_pos-2 := $n/preceding::w[position()=2][@c5='AJ0']

let $adj\_pos-3 := $n/preceding::w[position()=3][@c5='AJ0']

return

concat($n[@id]/string(@id),';',$n/string(),';',$adj\_pos-1/string(),';',$adj\_pos-2/string(),';',$adj\_pos-3/string())

## Returns all verb forms (of any type) occurring in a window of 1 to 3 words before an adverb particle (e.g., *up*, *out*):

for $avp in //w[@c5='AVP']

let $v\_pos1 := $avp/preceding::w[1][starts-with(@c5,'V')]

let $v\_pos2 := $avp/preceding::w[2][starts-with(@c5,'V')]

let $v\_pos3 := $avp/preceding::w[3][starts-with(@c5,'V')]

return

concat($v\_pos3/string(),';',$v\_pos2/string(),';',$v\_pos1/string(),';',$avp/string(),';',$avp[@id]/string(@id))

## Returns all words occurring in a window of 1 to 2 words left and right of a verb form (of any type):

for $v in //w[starts-with(@c5,'V')]

let $w\_left1 := $v/preceding::w[1]

let $w\_left2 := $v/preceding::w[2]

let $w\_right1 := $v/following::w[1]

let $w\_right2 := $v/following::w[2]

return

concat($w\_left2/string(),';',$w\_left1/string(),';',$v[@id]/string(@id),';',$v/string(),';',$w\_right1/string(),';',$w\_right2/string())

## Returns all adjectives occurring in a window of 1 to 2 words left and right of a noun (of any number and type):

for $n in //w[starts-with(@c5,'N')]

let $adj\_left1 := $n/preceding::w[1][@c5='AJ0']

let $adj\_left2 := $n/preceding::w[2][@c5='AJ0']

let $adj\_right1 := $n/following::w[1][@c5='AJ0']

let $adj\_right2 := $n/following::w[2][@c5='AJ0']

return

concat($adj\_left2/string(),';',$adj\_left1/string(),';',$n[@id]/string(@id),';',$n/string(),';',$adj\_right1/string(),';',$adj\_right2/string())

# Computing n-grams

## Produces an ordered list of all 3-grams:

for $ngram\_pos1 at $sequ in //w

let $ngram\_pos2 := $ngram\_pos1/following::w[position()=1]

let $ngram\_pos3 := $ngram\_pos1/following::w[position()=2]

return

concat($sequ,'. ', $ngram\_pos1/string(),' ', $ngram\_pos2/string(),' ', $ngram\_pos3/string())

Note: the sequence numbers are added by means of the keyword at followed by a variable name for sequence (here: at $sequ)

## Returns all 4-POS-grams:

for $ngram\_pos1 in //w[not(@c5='POS')]

let $ngram\_pos2 := $ngram\_pos1/following::w[position()=1]

let $ngram\_pos3 := $ngram\_pos1/following::w[position()=2]

let $ngram\_pos4 := $ngram\_pos1/following::w[position()=3]

return

concat($ngram\_pos1[@id]/string(@id),';',$ngram\_pos1[@c5]/string(@c5),';',$ngram\_pos2[@c5]/string(@c5),';',$ngram\_pos3[@c5]/string(@c5),';',$ngram\_pos4[@c5]/string(@c5))

# Constructing frequency lists

## Constructs a list of word types in descending order of frequency:

let $tokens :=

for $w in //w[not(@c5='POS')]

return

lower-case($w/string())

let $types := distinct-values($tokens)

for $type\_list in $types

let $freq := count($tokens[.=$type\_list])

where $freq >= 5

order by $freq descending

return

concat($type\_list,';',$freq)

## Constructs a list of POS tags in ascending order of frequency:

let $pos\_tokens:=

for $w in //w[@c5]

return

$w[@c5]/string(@c5)

let $types := distinct-values($pos\_tokens)

for $type\_list in $types

let $freq := count($pos\_tokens[.=$type\_list])

where $freq >= 5

order by $freq ascending

return

concat($type\_list,';',$freq)

## Constructs a list of word token bigrams in descending order of frequency:

let $tokens :=

for $w in //w

return

concat(lower-case($w/string()),' ',lower-case($w/following::w[1]/string()))

let $types := distinct-values($tokens)

for $type in $types

let $freq := count($tokens[.=$type])

where $freq >= 3

order by $freq descending

return

concat($type,';', $freq)

## Constructs a list of POS trigrams in descending order of frequency:

let $pos\_tokens:=

for $w in //w[@c5]

return

concat($w[@c5]/string(@c5),' ',$w[@c5]/following::w[1]/string(@c5),' ',$w[@c5]/following::w[2]/string(@c5))

let $types := distinct-values($pos\_tokens)

for $type\_list in $types

let $freq := count($pos\_tokens[.=$type\_list])

where $freq >= 5

order by $freq descending

return

concat($type\_list,';',$freq)

# Calculating positions

## Calculates positions of possessive pronouns (DPS):

for $whole\_doc in //wtext

let $words\_total := count($whole\_doc//w[not(@c5="POS")])

for $dps in //w[@c5='DPS']

let $words\_prec\_dps := count($dps/preceding::w[ancestor::wtext])

let $pos := $words\_prec\_dps div $words\_total

return

concat($pos,';',$dps/text())

## Calculates positions of anaphora; also returns corresponding values on 'repeat' attributes:

for $whole\_doc in //wtext

let $words\_total := count($whole\_doc//w[not(@c5="POS")])

for $anaph in //anaph

let $words\_prec := count($anaph/preceding::w[ancestor::wtext])

let $pos := $words\_prec div $words\_total

return

concat($anaph[@repeat]/string(@repeat),';',$pos)

## Calculates positions of first colons in tricolons:

for $whole\_doc in //wtext

let $words\_total := count($whole\_doc//w[not(@c5="POS")])

for $first.colon in //isoco[@ctype='tri'][@sequ='first']

let $words\_colon := count($first.colon//w[not(@c5="POS")])

let $words\_prec\_first.colon := count($first.colon/preceding::w[ancestor::wtext])

let $pos := $words\_prec\_first.colon div $words\_total

return

concat($words\_colon,';',$pos)

## Calculates positions of last colons:

for $whole\_doc in //wtext

let $words\_total := count($whole\_doc//w[not(@c5="POS")])

for $last.colon in //isoco[@ctype='di'][@sequ='second']

let $words\_colon := count($last.colon//w[not(@c5="POS")])

let $words\_prec\_last.colon := count($last.colon/preceding::w[ancestor::wtext])

let $pos := $words\_prec\_last.colon div $words\_total

return

concat($words\_colon,';',$pos)

Variable values in line 3: @ctype: 'di', 'tri', 'tet'; @sequ: 'second', 'third', 'fourth'

## Calculates positions of the last word in the last colon; also returns that last word and its 'c5' value:

for $whole\_doc in //wtext

let $words\_total := count($whole\_doc//w[not(@c5="POS")])

for $last\_colon in //isoco[@ctype='tet'][@sequ='fourth']

let $last\_word\_colon := $last\_colon/w[last()]

let $words\_prec\_last\_word\_colon := count($last\_word\_colon/preceding::w[ancestor::wtext])

let $pos := $words\_prec\_last\_word\_colon div $words\_total

return

concat($last\_word\_colon/string(),';',$last\_word\_colon[@c5]/string(@c5),';',$pos)

Variable values in line 3: @ctype: 'di', 'tri', 'tet'; @sequ: 'second', 'third', 'fourth'

# Constructing Key Word In Context (KWIC) displays

## Constructs a KWIC display for the search word “nation” in a 5-word window left and right of the node (L5-R5)

let $search\_word := 'our'

let $hits := //w[.=$search\_word]

let $window\_size := 5

for $hit at $sequ in $hits

let $left\_context :=

for $slot in reverse(1 to $window\_size)

let $left\_w := $hit/preceding::w[$slot]

return

$left\_w

let $right\_context :=

for $slot in (1 to $window\_size)

let $rw := $hit/following::w[$slot]

return

$rw

return

concat($sequ,'. ',string-join($left\_context/string(),' '), ' |', $hit/string(), '| ', string-join($right\_context/string(),' '))

Constructs a L3-R3 KWIC output for all adjectives (tagged 'AJ0') and all surrounding POS values:

let $search\_pos := //w[@c5='AJ0']

let $hits := //w[.=$search\_pos]

let $window\_size := 3

for $hit at $sequ in $hits

let $left\_context :=

for $slot in reverse(1 to $window\_size)

let $left\_w := $hit/preceding::w[$slot]

return

$left\_w

let $right\_context :=

for $slot in (1 to $window\_size)

let $rw := $hit/following::w[$slot]

return

$rw

return

concat($sequ,'. ',string-join($left\_context[@c5]/string(@c5),' '), ' |', $hit/string(), '| ', string-join($right\_context[@c5]/string(@c5),' '))

*First 4 lines of output:*

1. DPS |fellow| NN2 PNP VVB

2. NN1 PRP PNP |grateful| PRP AT0 NN1

3. PNP VHB VVN |mindful| PRF AT0 NN2

4. AV0 VVN AT0 |presidential| NN1 AT0 NN2