Appendix 1

dicta.py

```
#!/usr/local/bin/python3
#
# Paul Evans (10evans@cua.edu)
# 25 Oct -  8 Nov 2015
# 17 Oct - 24 Oct 2013
#
import re
import sys
def main():
    f = open('./edF.txt', 'r')
    file = f.read()
    toc = open('./toc_all.txt', 'r')
    dictionary_Fr = {} # Friedberg
    dictionary_1r = {} # first recension
    dictionary_2r = {} # second recension
    # (?<=...) positive lookbehind assertion.
    dicta = re.findall('(?:\<T [AP]\>|(?<=\<T [AP]\>))(.*?)'    # dictum starts with dictum ante or dictum
post tag.
        '(?:'                   # non-capturing group.
            '\<1 [CD][CP]?\>|'  # dictum ends with major division,
            '\<2 \d{1,3}\>|'    # or number of major division,
            '\<3 \d{1,2}\>|'    # or number of question,
            '\<4 \d{1,3}\>|'    # or number of canon,
            '\<P 1\>|'          # or Palea,
            '\<T [AIPRT]\>'      # or inscription or text tag.
        ')', file, re.S)        # re.S (re.DOTALL) makes '.' special character match any character
including newline.
    print('expected 1277 dicta, found ' + str(len(dicta)) + ' dicta', file=sys.stderr)
```

```
    for dictum in dicta:
        dictum = re.sub('\<S \d{1,4}\>\<L 1\> \-\d{1,4}\+', '', dictum) # remove page and line number tags.
        dictum = re.sub('\<P 1\> \-\[PALEA\.\+', '', dictum)    # remove Palea tags.
        dictum = re.sub('\-.*?\+', '', dictum)
        dictum = re.sub(re.compile('\-\[.*?\]\+', re.S), '', dictum)
        dictum = re.sub('\s+', ' ', dictum)
        dictum = re.sub('^\s+', '', dictum) # remove leading whitespace characters
        dictum = re.sub('\s+$', '', dictum) # remove trailing whitespace characters
        key = toc.readline().rstrip()
        if key in dictionary_Fr:
        # if there's already a dictionary entry with this key, merge the entries
            # print('duplicate key: ' + key, file=sys.stderr)
            dictum = dictionary_Fr[key] + ' ' + dictum
        dictionary_Fr[key] = dictum

    keys = tuple(open('./toc_1r.txt', 'r'))
    for key in keys:
        key = key.rstrip()
        dictionary_1r[key] = dictionary_Fr[key] # copy dictum from Friedberg dictionary into first-
recension dictionary

    keys = tuple(open('./toc_2r.txt', 'r'))
    for key in keys:
        key = key.rstrip()
        dictionary_2r[key] = dictionary_Fr[key] # copy dictum from Friedberg dictionary into second-
recension dictionary

    keysandpatterns = [
        {'key': 'D.25 d.p.c.1', 'pattern': '(Ex hac epistola liquet, quid cuiusque offitii sit\.)'},
        {'key': 'D.25 d.p.c.3', 'pattern': '(Nunc autem per.*?mentem eius remordeat\.)'},
        {'key': 'D.26 d.p.c.4', 'pattern': '(Iohannes etiam Baptista.*?alteram habuisse probantur\.)'},
        {'key': 'D.30 d.a.c.1', 'pattern': '(Illud autem, quod.*?que coniugium detestabatur,)'},
        {'key': 'D.31 d.p.c.11', 'pattern': '(Ut igitur ex.*?reddere non ualent. Sed obicitur illud
```

```
Tripartitae ystoriae:)'},
        {'key': 'D.42 d.p.c.1', 'pattern': '(Hinc etiam Iohannes.*?de ecclesia eiciebat\.)'},
        {'key': 'D.45 d.p.c.17', 'pattern': '(Hinc etiam alibi.*?uero patrem exhibeat.")'},
        {'key': 'D.47 d.p.c.8', 'pattern': '(Necesse est etiam.*?sollicitam diligentiam exhibebit\?)'},
        {'key': 'D.54 d.p.c.23', 'pattern': '(Ecce, quomodo serui.*?quomodo non admittantur\.)'},
        {'key': 'D.63 d.p.c.28', 'pattern': '(Verum, quia inperatores.*?anathematis uinculo
innodaretur,)(.*?)(Postremo presentibus legatis.*?ecclesiae Dei conferentes\.)'},
        {'key': 'D.63 d.p.c.34', 'pattern': '(Ex his constitutionibus.*?habita constitutum est\.)'},
        {'key': 'D.68 d.p.c.2', 'pattern': '(Quod ergo consecratus.*?ad cautelam salutis\.)'},
        {'key': 'C.1 q.1 d.p.c.51', 'pattern': '(Sed notandum est.*?Non sanat baptismus perfidorum,
etc.")'},
        {'key': 'C.1 q.1 d.p.c.123', 'pattern': '(Quolibet ergo munere.*?falsa diiudicatur ordinatio\.)'},
        {'key': 'C.1 q.4 d.p.c.9', 'pattern': '(Cum ergo de baptizatis.*?impediat nomen erroris\.)'},
        {'key': 'C.1 q.4 d.p.c.12', 'pattern': '(Ignorabat autem Petrus.*?permittitur ignorare, aliis
non\.)'},
        # {'key': 'C.2 q.3 d.p.c.7', 'pattern': '(Notandum quoque est.*?quod obiecerat desistat\.)'}, # @.2
        {'key': 'C.2 q.3 d.p.c.7', 'pattern': '(Notandum quoque est.*?in Libro Capitulorum:)'}, # @.2
        {'key': 'C.2 q.6 d.p.c.31', 'pattern': '(Forma uero appellationis.*?in scriptis fieri debent\.)'},
        {'key': 'C.2 q.6 d.p.c.39', 'pattern': '(Cum autem in.*?suam agere oportet\.)'},
        {'key': 'C.2 q.7 d.p.c.40', 'pattern': '(Cum ergo Petrus.*?suscipere reprehensionem
subditorum\.)'},
        {'key': 'C.2 q.8 d.p.c.5', 'pattern': '(Sed Calixtus Papa.*?per epistolam accusare audeat\.)'},
        {'key': 'C.3 q.1 d.p.c.6', 'pattern': '(Patet ergo, quod.*?quam uocentur ad causam\.)'},
        {'key': 'C.3 q.11 d.p.c.3', 'pattern': '(Hoc autem intelligendum.*?auctoritatibus non
prohibetur\.)'},
        # {'key': 'C.4 q.2 d.p.c.3', 'pattern': '(Sed obicitur illud.*?humanae actionis trahenda\.)'},
        {'key': 'C.5 q.3 d.p.c.1', 'pattern': '(Ecce episcopus.*?se agere licet\.)'},
        {'key': 'C.6 q.1 d.p.c.21', 'pattern': '(Verum hoc Augustini.*?accusatione ipse repellit\.)'},
        {'key': 'C.11 q.1 d.p.c.34', 'pattern': '(Non ait propter.*?quam criminalem intelligens\.)'},
        {'key': 'C.11 q.3 d.p.c.40', 'pattern': '(Premissis auctoritatibus, quibus.*?in se exceperunt\.)'},
        {'key': 'C.13 q.1 d.p.c.1', 'pattern': '(In diocesi autem.*?qui secum erant\.)(.*?)(Quia ergo
nos.*?ad diocesianum transferre\.)'},
        {'key': 'C.13 q.2 d.p.c.3', 'pattern': '(Item Ioseph, moriens.*?eo sepultus est\?)(.*?)(Exemplo
igitur istorum.*?uoluntate tumulandi consistit\.)'},
```

```
        {'key': 'C.13 q.2 d.p.c.8', 'pattern': '(Hac nimirum auctoritate.*?quam prohibetur
transscendere\.)'},
        {'key': 'C.14 q.1 d.p.c.1', 'pattern': '(Quia ergo generaliter.*?prohibentur stare coram
iudice\.)'}, # 'stare coram iudice' occurs twice
        {'key': 'C.14 q.2 d.p.c.1', 'pattern': '(Potest etiam intelligi.*?pauperum, testimonium
dicant\.)'},
        {'key': 'C.14 q.5 d.p.c.14', 'pattern': '(Sed hoc multipliciter.*?bonum possunt conuerti\.)'},
        {'key': 'C.15 q.1 d.p.c.3', 'pattern': '(Ex eo autem.*?penam aut gloriam.")'},
        {'key': 'C.15 q.1 d.p.c.11', 'pattern': '(Cum itaque qui.*?Obicitur autem)'},
        {'key': 'C.15 q.1 d.p.c.12', 'pattern': '(Sunt quedam, que.*?muneris executionem inpediunt\.)'},
        {'key': 'C.16 q.1 d.p.c.40', 'pattern': '(.*?)(Ostendit ergo Ieronimus.*?ipsum inperfectis
connumerans\.)(.*?)(Ecce sufficienter monstratum.*?assecuntur potestatis executionem\.)(.*?$)'}, #
inperfectis
        {'key': 'C.16 q.1 d.p.c.47', 'pattern': '(Quod autem dicitur.*?duos potest diuidi,)'},
        {'key': 'C.16 q.1 d.p.c.53', 'pattern': '(Sicut duo episcopatus.*?ad paucitatem redigeretur\.)'},
        {'key': 'C.16 q.3 d.p.c.15', 'pattern': '(Potest etiam aliter.*?obici non potest\.)'},
        {'key': 'C.16 q.3 d.p.c.16', 'pattern': '(Sed sola prescriptione.*?spatio prescribi possunt\.)'},
        {'key': 'C.17 q.2 d.p.c.2', 'pattern': '(Ecce iste se.*?concepit, et ore pronunciauit\.)'}, # 'et
ore pronunciauit' occurs twice
        {'key': 'C.21 q.2 d.p.c.3', 'pattern': '(Sed aliud est.*?omnibus modis prohibetur\.)'},
        {'key': 'C.22 q.1 d.p.c.16', 'pattern': '(Sic etiam cum.*?creatorem iurat mendaciter\.)'},
        {'key': 'C.22 q.2 d.p.c.5', 'pattern': '(Ille ergo falsum.*?esse quod iurat\.)'},
        {'key': 'C.23 q.4 d.p.c.26', 'pattern': '(Potest in hac.*?personae quendam excommunicauerat,)'},
        {'key': 'C.23 q.4 d.p.c.27', 'pattern': '(ostendens, quod peccata.*?potius dissimulanda sunt)'},
        {'key': 'C.23 q.4 d.p.c.30', 'pattern': '(Quod autem peccatum.*?patienter tollerasse
asseritur\.)'},
        {'key': 'C.23 q.8 d.p.c.25', 'pattern': '(Hinc datur intelligi.*?Pontificis fieri debet\.)'},
        {'key': 'C.23 q.8 d.p.c.27', 'pattern': '(Reprehenduntur ergo Gallicani.*?orationibus Deo
conmendent\.)'},
        {'key': 'C.29 q.1 d.a.c.1', 'pattern': '(Quod autem coniugium.*?potest eam dimittere,)'},
        {'key': 'C.29 q.2 d.p.c.6', 'pattern': '(Cum dicitur: "sciens.*?fraude decepta est;)'},
        {'key': 'C.30 q.4 d.p.c.5', 'pattern': '(Notandum uero est.*?uiro suo cognoscitur\.)'},
        {'key': 'C.31 q.1 d.p.c.7', 'pattern': '(Sed obicitur: Dauid.*?quam significatione futurorum\.)'},
        {'key': 'C.32 q.1 d.p.c.10', 'pattern': '(Si ergo, ut.*?sed adulteri appellantur\.)'},
```

```
        {'key': 'C.32 q.4 d.p.c.10', 'pattern': '(Ecce, quod nullo.*?nomine iudicantur indigni\.)'},
        {'key': 'C.33 q.2 d.p.c.9', 'pattern': '(In premissis auctoritatibus.*?eis misericordia
inpendatur\.)'},
        {'key': 'de Pen. D.1 d.a.c.1', 'pattern': '(Utrum sola cordis.*?promereri, iuxta illud)'}, #
d.a.c.1
        {'key': 'de Pen. D.1 d.p.c.87', 'pattern': '(His auctoritatibus asseritur.*?iugiter confiteri
debemus\.)(.*?)(Similiter et illud.*?de penitencia ait:)'},
        {'key': 'de Pen. D.2 d.a.c.1', 'pattern': '(Alii dicunt penitenciam.*?tibi aliquid contingat.")'},
        {'key': 'de Pen. D.2 d.p.c.24', 'pattern': '(Hec itaque karitas.*?redeunt et cetera.")'},
        {'key': 'C.35 q.2 d.p.c.21', 'pattern': '(Hac auctoritate dum.*?ducat in uxorem.")'}, # ducat
    ]

    for i in range (len(keysandpatterns)):
        key = keysandpatterns[i]['key']
        pattern = keysandpatterns[i]['pattern']
        result = re.search(pattern, dictionary_Fr[key])
        if result:
            if len(result.groups()) == 1:
                dictionary_1r[key] = fixString(result.group(1))
                dictionary_2r[key] = fixString(re.sub(pattern, '', dictionary_2r[key]))
            elif len(result.groups()) == 3:
                dictionary_1r[key] = fixString(result.group(1)) + ' ' + fixString(result.group(3))
                dictionary_2r[key] = fixString(result.group(2))
            elif len(result.groups()) == 5: # C.16 q.1 d.p.c.40
                dictionary_1r[key] = fixString(result.group(2)) + ' ' + fixString(result.group(4))
                dictionary_2r[key] = fixString(result.group(1)) + ' ' + fixString(result.group(3)) + ' ' +
fixString(result.group(5))
        else:
            print('no match: ' + key + '\n' + dictionary_Fr[key], file=sys.stderr)

    # insert
    key = 'C.3 q.1 d.p.c.2'
    dictionary_1r[key] = '''Sed notandum est quod restitutio alia fit per presentiam iudicis, ueluti cum
```

```
dicitur a iudice: "Censeo te in integrum restituendum", qua restitutione animo tantum, non corpore
possessio recipitur. Alia fit per executorem iudicis quando restitutus corporalem recipit possessionem.
Queritur ergo que harum concedatur expoliatis, an illa tantum, que fit per sententiam iudicis, an illa
etiam que fit per executorem sententiae, qua expoliatis presentialiter omnia reciduntur. Hec ultima
expoliatis prestanda est.'''
    # append
    key = 'C.3 q.1 d.p.c.6'
    dictionary_1r[key] = dictionary_1r[key] + ''' His ita respondetur. Si uicium electionis ecclesie notum
fuerit et ideo reprobati fuerint et si aliqua uiolentia in sedibus illis irrepserit eiecti restitutionem
postulare non possunt. Si autem ecclesia eos per pacientiam tolerare uoluerit et eis gradum honoris
concesserit et si uiciosa fuerit eorum electio, tamen post eiectionem restituendi sunt, ante regularem ad
synodi uocationem.'''
    # special fix
    key = 'C.15 q.1 d.p.c.11'
    dictionary_1r[key] = dictionary_1r[key][0:-1] + ':'
    # interpolate
    key = 'C.15 q.3 d.p.c.4'
    pattern = '(Cum autem sacris.*?hoc non infertur\.)(.*?)(Quamuis igitur sacris.*?credi non
oportet\.)(.*?$)'
    result = re.search(pattern, dictionary_1r[key])
    if result:
        dictionary_1r[key] = fixString(result.group(1)) + ''' Quecumque enim persone humanis legibus
copulari prohibentur et diuinis, non omnium copula a sacris canonibus admittitur, quorum conuentio legibus
imperatorum indulgetur. ''' + fixString(result.group(3))
        dictionary_2r[key] = fixString(result.group(2)) + ' ' + fixString(result.group(4))
    else:
        print('no match: ' + key + '\n' + dictionary_1r[key], file=sys.stderr)
    # insert
    key = 'C.21 q.3 d.a.c.1'
    dictionary_1r[key] = '''Quod autem clerici secularium negotiorum procuratores esse non ualeant
auctoritate Calcedonensis synodi probatur in qua sic statutum est legitur:'''
    # append
    key = 'C.23 q.8 d.p.c.25'
    dictionary_1r[key] = dictionary_1r[key] + ''' Unde in quodam concilio statutum est ut episcopi non
```

```
proficiscantur ad comitatum nisi formatas ab apostolico acceperint.'''
    # append
    key = 'de Pen. D.1 d.a.c.1'
    dictionary_1r['de Pen. D.1 d.a.c.1'] = dictionary_1r[key].rstrip('.') + ''' Leonis pape:'''

    all = open('./Gratian1.txt', 'w')
    keys = tuple(open('./toc_1r.txt', 'r'))
    for key in keys:
        key = key.rstrip()
        outfilename = './1r/' + key + '.txt'
        each = open(outfilename, 'w')
        each.write(dictionary_1r[key] + '\n')
        all.write(dictionary_1r[key] + '\n')
        each.close
    all.close()

    all = open('./Gratian2.txt', 'w')
    keys = tuple(open('./toc_2r.txt', 'r'))
    for key in keys:
        key = key.rstrip()
        outfilename = './2r/' + key + '.txt'
        each = open(outfilename, 'w')
        each.write(dictionary_2r[key] + '\n')
        all.write(dictionary_2r[key] + '\n')
        each.close
    all.close()

def fixString(string):
    string = re.sub('\s+', ' ', string) # 2r
    string = re.sub('^\s+', '', string) # 2r
    string = re.sub('\s+$', '', string) # 2r
    if string[-1] == ',' or string[-1] == ';':
        string = string[0:-1] + '.'
```

```
    if string[-1].isalpha():
        string = string + '.'
    return string

if __name__ == '__main__':
    main()
```

Appendix 2

main.py

```
#!/usr/local/bin/python3
#
# Paul Evans (10evans@cua.edu)
# 8 February 2015 -
# 12 February 2015
#
import re
import sys
import parse
def main():
    file = open('./edF.txt', 'r').read()
    decretum = parse.parse_all(preprocess(file))
    traverse(decretum[0])
    traverse(decretum[1])
    traverse(decretum[2])

def traverse(tree):
    for i in range(len(tree[1])):
        subtree = tree[1][i]
        if isinstance(subtree[1], list):
            tag = subtree[0]
            print(tag)
            traverse(subtree)
        elif isinstance(subtree[1], str):
            tag = subtree[0]
            text = subtree[1]
            print(tag, text)
    return
```

```
def preprocess(text):
    text = re.sub(re.compile('\-.*?\+', re.S), '', text) # remove comments
    text = re.sub('\<S \d{1,4}\>', '', text) # remove page number tags
    text = re.sub('\<L \d{1,2}\>', '', text) # remove line number tags
    text = re.sub('\<P 1\>|\<P 0\>', '', text) # remove Palea tags
    text = re.sub('\s+', ' ', text) # remove multiple whitespace characters
    text = re.sub('\s+$', '', text) # remove trailing whitespace characters
    return(text)

if __name__ == '__main__':
    main()
```

parse.py

```
#!/usr/local/bin/python3
#
# Paul Evans (10evans@cua.edu)
# 23 January 2015 -
# 12 February 2015
#
import re
import sys
def parse_all(text):
    part_list = []
    m = re.search('(\<1 D\>.*?)(\<1 C\>.*?)(\<1 DC\>.*?)$', text, re.S)
    part_list.append(('<1 D>', parse_part_1(m.group(1))))
    part_list.append(('<1 C>', parse_part_2(m.group(2))))
    part_list.append(('<1 DC>', parse_part_3(m.group(3))))
    return(part_list)

# D.1-101
def parse_part_1(text):
    distinction_list = []
```

```
    distinctions = re.findall('(?:\<1 D\>)(.*?)(?=\<1 D\>|$)', text)
    for distinction in distinctions:
        distinction = distinction.strip(' ')
        m = re.match('(\<2 \d{1,3}\>) (\<T A\>) (.*?) (\<4 1\>.*?)$', distinction)
        tag = m.group(1)
        node = (m.group(2), m.group(3)) # d.a.c.1 tag-text tuple
        canon_list = parse_canons(m.group(4))
        canon_list.insert(0, node)
        distinction_list.append((tag, canon_list))
    return(distinction_list)

# C.1-36
def parse_part_2(text):
    case_list = []
    cases = re.findall('(?:\<1 C\>)(.*?)(?=\<1 C\>|$)', text)
    for case in cases:
        case = case.strip(' ')
        m = re.match('(\<2 \d{1,2}\>)(\<T Q\>) (.*?) (\<3 1\>.*?)$', case)
        tag = m.group(1)
        node = (m.group(2), m.group(3)) # d.init. tag-text tuple
        question_list = parse_questions(m.group(4))
        question_list.insert(0, node)
        case_list.append((tag, question_list))
    return(case_list)

# de Consecratione
def parse_part_3(text):
    distinction_list = []
    distinctions = re.findall('(?:\<1 DC\>)(.*?)(?=\<1 DC\>|$)', text)
    for distinction in distinctions:
        distinction = distinction.strip(' ')
        m = re.match('(\<2 \d\>) (\<4 1\>.*?)$', distinction)
        tag = m.group(1)
```

```
        canon_list = parse_canons(m.group(2))
        distinction_list.append((tag, canon_list))
    return(distinction_list)

def parse_questions(text):
    question_list = []
    questions = re.findall('(\<3 \d{1,2}\>.*?)(?=\<3 \d{1,2}\>|$)', text)
    for question in questions:
        question = question.strip(' ')
        m0 = re.match('(\<3 \d{1,2}\>) (\<T A\>) (.*?) (\<1 DP\>.*?)$', question) # C.33 q.3 (de Pen.)
        m1 = re.match('(\<3 \d{1,2}\>) (\<T A\>) (.*?) (\<4 1\>.*?)$', question)
        m2 = re.match('(\<3 \d{1,2}\>) (\<T A\>) (.*?)$', question) # C.11 q.2, C.17 q.3, C.22 q.3, C.29
q.1
        if m0:
            tag = m0.group(1)
            node = (m0.group(2), m0.group(3)) # d.a.c.1 tag-text tuple
            distinction_list = parse_de_pen(m0.group(4))
            question_list.append((tag, [node, ('<1 DP>', distinction_list)]))
        elif m1:
            tag = m1.group(1)
            node = (m1.group(2), m1.group(3)) # d.a.c.1 tag-text tuple
            canon_list = parse_canons(m1.group(4))
            canon_list.insert(0, node)
            question_list.append((tag, canon_list))
        elif m2:
            tag = m2.group(1)
            node = (m2.group(2), m2.group(3)) # d.a.c.1 tag-text tuple
            question_list.append((tag, [node]))
    return(question_list)

# de Penitentia
def parse_de_pen(text):
    distinction_list = []
```

```
    distinctions = re.findall('(?:\<1 DP\>)(.*?)(?=\<1 DP\>|$)', text)
    for distinction in distinctions:
        distinction = distinction.strip(' ')
        m = re.match('(\<2 \d\>) (\<T A\>) (.*?) (\<4 1\>.*?)$', distinction)
        tag = m.group(1)
        node = (m.group(2), m.group(3)) # d.a.c.1 tag-text tuple
        canon_list = parse_canons(m.group(4))
        canon_list.insert(0, node)
        distinction_list.append((tag, canon_list))
    return(distinction_list)

# return list of canons
def parse_canons(text):
    canon_list = []
    canons = re.findall('(\<4 \d{1,3}\>.*?)(?=\<4 \d{1,3}\>|$)', text)
    for canon in canons:
        canon = canon.strip(' ')
        m = re.match('(\<4 \d{1,3}\>) (.*?)$', canon)
        if m:
            nodes = parse_nodes(m.group(2))
        else: # C.1 q.4 c.6
            m = re.match('(\<4 \d{1,3}\>)$', canon)
            nodes = []
        canon_list.append((m.group(1), nodes))
    return(canon_list)

# return list of terminal nodes (tag-text tuples)
def parse_nodes(text):
    node_list = []
    nodes = re.findall('(\<T [AIPRT]\>.*?)(?=\<T [AIPRT]\>|$)', text)
    for node in nodes:
        node =node.strip(' ')
        m = re.match('(\<T [AIPRT]\>) (.*?)$', node)
```

```
            node_list.append((m.group(1), m.group(2)))
        return(node_list)

if __name__ == '__main__':
    main()
```