Edit Distance and sequence alignment

Xavier Dupré
http://www.xavierdupre.fr/

3 mai 2013

Résumé

Correction

0.1 Code for the edit distance which returns an alignment

```
ïż£
class EditDistanceAlignment_Element :
    def __init__ (self, pos, cost, el1, el2, op) :
        self.pos  = pos
        self.el1  = el1
        self.el2  = el2
        self.cost = cost
        self.op   = op    # "cmp", "ins1", "ins2", None

    def __str__ (self) :
        return "pos1=%d pos2=%d cost=%s op=%s el1=%s el2=%s" % \
            (self.pos[0], self.pos[1], "%1.3f" % self.cost if self.cost != None else "None",
                    self.op, self.el1, self.el2)

def EditDistanceAlignment ( exp1,
                            exp2,
                            cmpinsFunction      = lambda x,y : 0.0 if x == y else 1.0,
                            constantEpsilon     = 1e-5,
                            constantInfinite    = 1e5,
                            setPosition         = [ (-1,-1), (-1,0), (0,-1), ],
                            returnDistance      = True) :

    if len(exp1) == 0 :
        return 0 if len(exp2) == 0 else sum ( [ cmpinsFunction (None, e) for e in exp2] )
    if len(exp2) == 0 :
        return sum ( [ cmpinsFunction (e, None) for e in exp1] )

    l1   = len(exp1) + 1
    l2   = len(exp2) + 1
    l    = l1*l2
    pa   = { }
    pred = { }

    for n,el in enumerate (exp1) :
```

```
    pa  [n,0] = cmpinsFunction(el,None)
    pred[n,0] = (n-1,0)

for n,el in enumerate (exp2) :
    pa  [0,n] = cmpinsFunction(None,el)
    pred[0,n] = (0,n-1)

pa[0,0] = 0.0

for n1,el1 in enumerate (exp1) :
    for n2,el2 in enumerate (exp2) :

        cost = [ ]
        for dx,dy in setPosition :
            c = pa.get ( (n1+dx+1, n2+dy+1), constantInfinite)
            if dx == 0 :
                if dy == -1 :
                    cost.append ( (c+cmpinsFunction (None, el2), (dx,dy)  ) )
                else :
                    raise Exception ("unable to deal with with case yet %d,%d" %(dx,dy))
            elif dy == 0 :
                if dx == -1 :
                    cost.append ( (c+cmpinsFunction (el1, None), (dx,dy)  ) )
                else :
                    raise Exception ("unable to deal with with case yet %d,%d" %(dx,dy))
            else :
                if dx == -1 and dy == -1:
                    cost.append ( (c+cmpinsFunction (el1, el2), (dx,dy)  ) )
                else :
                    raise Exception ("unable to deal with with case yet %d,%d" %(dx,dy))

            mn               = min ( cost )
            pa   [n1+1,n2+1] = mn[0]
            pred [n1+1,n2+1] = (n1+mn[1][0]+1,n2+mn[1][1]+1)

if returnDistance :
    return pa [ l1-1,l2-1 ]
else :
    align = [ ]
    p     = l1-1,l2-1
    while p[0] != -1 and p[1] != -1 :
        e = EditDistanceAlignment_Element (p, pa[p], None,None,None)
        align.append (e)
        p = pred[p]

    align.pop()
    align.reverse()

    for n,th in enumerate(align) :
        pr      = None if n == 0 else align[n-1]
        th.pos  = (th.pos[0]-1,th.pos[1]-1)

        if pr == None :
            if  th.pos == (0,0) : kind = "cmp"
            elif th.pos[1] == -1 : kind = "ins1"
            else : kind = "ins2"
        else :
            if th.pos == (pr.pos[0]+1, pr.pos[1]+1) : kind = "cmp"
            elif th.pos[0] == pr.pos[0] : kind = "ins2"
```

```
                else : kind = "ins1"

            th.op = kind
            if kind == "cmp" :
                th.el1 = exp1[th.pos[0]]
                th.el2 = exp2[th.pos[1]]
            elif kind == "ins1" :
                th.el1 = exp1[th.pos[0]]
                th.el2 = None
            elif kind == "ins2" :
                th.el1 = None
                th.el2 = exp2[th.pos[1]]

        return align

def EditDistanceAlignmentNormalized ( exp1,
                            exp2,
                            cmpinsFunction      = lambda x,y : 0.0 if x == y else 1.0,
                            constantEpsilon     = 1e-5,
                            constantInfinite    = 1e5,
                            setPosition         = [ (-1,-1), (-1,0), (0,-1), ],
                            returnDistance      = True) :

    def function (x,y) :
        if x == y : return 0
        l1 = 0 if x == None else len(x)
        l2 = 0 if y == None else len(y)
        if l1 > l2 :
            return l2*(0.5 / len(exp1) + 0.5 / len(exp2)) + (l1-l2)*1.0 / len(exp1)
        elif l2 > l1 :
            return l1*(0.5 / len(exp1) + 0.5 / len(exp2)) + (l2-l1)*1.0 / len(exp2)
        else :
            return l1*(0.5 / len(exp1) + 0.5 / len(exp2))

    return EditDistanceAlignment(exp1, exp2, function,
                                 constantEpsilon, constantInfinite, setPosition, returnDistance)

def EditDistanceAlignmentWordSequenceInString (exp1,
                            exp2,
                            cmpinsFunction      = lambda x,y : 0.0 if x == y else 1.0,
                            constantEpsilon     = 1e-5,
                            constantInfinite    = 1e5,
                            setPosition         = [ (-1,-1), (-1,0), (0,-1), ],
                            returnDistance      = True) :

    spl1 = exp1.split(' ')
    spl2 = exp2.split(' ')
    def function (x,y) :
        if x == y : return 0
        if x == None : return len(spl2)
        if y == None : return len(spl1)
        return EditDistanceAlignment(x,y, cmpinsFunction,
                            constantEpsilon, constantInfinite,
                            setPosition, True)

    return EditDistanceAlignment(spl1, spl2, function,
                                 constantEpsilon, constantInfinite, setPosition, returnDistance)

if __name__ == "__main__" :
```

```
def cmpins (x,y) :
    if x == y : return 0
    if x in ("n","m") and y in ("n","m") : return 0.25
    if (x == "s" and y == None) or (x == None and y == "s") : return 0.5
    return 1

# alignment
res = EditDistanceAlignment ("cmp", "cmps", cmpins, returnDistance = False)
for _ in res : print (_)

# only distance
res = EditDistanceAlignment ("cmp", "cmps", cmpins)
print (res)

# alignment
# other writing
res = EditDistanceAlignment (list("cmp"), list("cmps"), cmpins, returnDistance = False)
for _ in res : print (_)

# numbers
def cmpins_int (x,y) :
    if x == y : return 0.
    if x in (8,9) and y in (8,9) : return 0.25
    return 1.

res = EditDistanceAlignment ([1,2,8,7], [1,2,9,7], cmpins_int)
print (res)
res = EditDistanceAlignment ([1,2,8,7], [1,2,9,7], cmpins_int, returnDistance = False)
for _ in res : print (_)
```