SLP Assignment E September 22, 2020

Information Retrieval

In this assignment you will build the core of a small domain-specific search engine. Luckily the crawl has been done for you, speed is not required, and no interface is needed. Thus you will only need to implement the similarity computation, design and implement some feature(s), and do some experiments.

Specifically, you will process queries about UTEP Computer Science faculty. The collection of documents is csFaculty.txt, sample queries are in trainingQueries.txt, and the final set for evaluation is testQueries.txt. You may also use the skeleton code, irStub.py. These are all available on the course website.

Do at least four experiments. You almost certainly will want to try word unigrams instead of character trigrams, and to implement equation 6.10. Also consider trying any of the techniques mentioned in the textbook (tf-idf, better length normalization, a stoplist, stemming, synsets, etc.), or something original.

Work in pairs.

Turn in a report structured similarly to the one for the previous assignments, including explicit failure analysis.

Due October 8. 15 points.

csFaculty.txt (sample)

MONIKA AKBAR Assistant Professor, Computer Science - Cyber-Share Dr. Monika Akbar is a computer scientist, Research Assistant Professor, and Assistant Director at Cyber-Share. Her research interests include data modeling and representation, digital libraries, online communities, data mining, and CS education. Over the years, she has worked on digital library innovations to assist computing educators. Incorporating online communities into digital libraries and engaging users in participating in those communities were two of the foci of her PhD. Her current research looks at developing a robust theoretical framework for data and information management systems that supports data analytics. She is an active member of associations that work on broadening participation and increasing the number of female students in Computing. KEYWORDS CS education Data analytics Data and information management information Security Smart transportation UTEP STRATEGIC AREAS Education for the 21st Century Demographic Cross-cutting: Cyberinfrastructure and Collaborative Environments Cross-cutting: Emerging Technologies: Information Technology, Biotechnology & Nanotechnology Other

akbar

News !! Our research “Reducing UML Modeling Tool Complexity with Architectural Contexts and Viewpoints“ won the Best Paper Award at MODELSWARD 2018, with collaborators from French Alternative Energies and Atomic Energy Commission (CEA LIST) and Modelware Solutions. The paper introduces a radical approach to reducing design complexity supported by a working implementation in Papyrus, the Flagship Modeling Platform. ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS)2018 Paper Accepted!! Pre-print available here. badreddin%2c-omar Dr. Omar Badreddin News!! Paper Accepted, 7th International Symposium on Leveraging Applications of Formal Methods, Verification & Validation (ISOLA 2016). “Modeling Meets Programming: A Comparative Study in Model Driven Engineering Action Languages”. [Full Text] Dr. Badreddin is the author of Susereum. He is a researcher in the field of Sustainable Software Engineering. The research is focused on exploring novel methods to reducing software complexity often by innovating novel design approaches. Our research team evaluates existing design languages and investigates how they affect design activities for medium and large software systems. Those investigations drive the exploration for novel approaches for software design. Many of the research findings have made its way into prominent software design tools including Papyrus, WebSphere Monitor, and others. Model Driven Software Engineering Various flavors of modeling and levels of formalisms for Designs Reducing complexity in large software systems can have broad implications well beyond the software engineering field. Our research team applies many of the research findings to Healthcare Informatic challenges in hospitals in the U.S and Canada. We have active collaborations with hospitals in North America, focusing on exploring techniques for compliance and clinical pathway management. Dr. Badreddin’s main research areas include Cyber-Physical Systems Design and Testing, Model-Driven Engineering, and Model-Driven Systems Engineering. His collaborators include NASA Jet Propulsion Lab, French Alternative Energies and Atomic Energy Commission, and IBM. His recent work is related to testing untestable systems. Untestable systems are those that cannot be tested due to the cost of performing tests (i.e collision-avoidance systems), or due to difficulties in sitting the environment for test (i.e deep ocean explorers).

badreddin

trainingQueries.txt (sample)

Internet of Things

salamah acosta freudenthal

Mobile Netorks

freudenthal salamah tosh

Systems

freudenthal salamah tosh

research like Alan Black

ward novick fuentes

Reliable Systems

freudenthal ceberio salamah

```
# Nigel Ward, UTEP, October 2018; Speech and Language Processing;  Assignment E: Information Retrieval
# This is just a skeleton that needs to be fleshed out. It is not intended as an example of good Python style

def parseAlternatingLinesFile(file):     #-----------------------------
   # read a sequence of pairs of lines, e.g. text of webpage(s), name/URL
   sequenceA = []
   sequenceB = []
   fp = open(file, 'r')
   expectingA = True
   for line in fp.readlines():
       if expectingA:
           sequenceA.append(line.rstrip())
           expectingA = False
       else:
           sequenceB.append(line.rstrip())
           expectingA = True
   fp.close()
   return sequenceA, sequenceB

def characterTrigrams(text):         #----------------------------
  return [text[i:i+3] for i in range(len(text)-3+1)]

def computeFeatures(text, trigramInventory):        #-----------------------------
    # catches the similarities between  "social" and "societal" etc.
    # but really should be replaced with something better
    trigrams = characterTrigrams(text)
    counts = {}
    for trigram in trigrams:
        if trigram in trigramInventory:
            if trigram in counts:
                counts[trigram] += 1
            else:
                counts[trigram] = 1
    return counts

def computeSimilarity(dict1, dict2):   #-----------------------------
    # ad hoc and inefficient
    matchCount = 0
    for tri in dict1:
        if tri in dict2:
            #print "match on " + tri
            matchCount += 1
    similarity = matchCount / float(len(dict2))
    #print 'similarity %.3f' % similarity
    return similarity

def retrieve(queries, trigramInventory, archive):      #-----------------------------
    # returns a 2-D array: for each query, the top 3 results found
    top3sets = []
    for query in queries:
        q = computeFeatures(query, trigramInventory)
        #print 'query is %s, query features are %s ' % (query, q)
        similarities = [computeSimilarity(q,d) for d in archive]
        top3indices = np.argsort(similarities)[0:3]
        #print "top three indices are %s " % top3indices
        top3sets.append(top3indices)
    return top3sets

def valueOfSuggestion(result, position, targets):   #-----------------------------
    weight = [1.0, .5, .25]
    if result in targets:
        return weight[max(position, targets.index(result))]
    else:
        return 0

def scoreResults(results, targets):   #-----------------------------
    return sum([valueOfSuggestion(results[i], i, targets) for i in [0,1,2]])

def scoreAllResults(queries, results, targets, descriptor):   #-----------------------------
    print '\nScores for ' + descriptor
    scores = []
    for q, r, t in zip(queries, results, targets):
        s = scoreResults(r, t)
        #print 'for query %s, results = %s, targets = %s, score = %.3f' % (q, r, t, s)
        scores.append(s)
    overallScore = np.mean(scores)
    print 'all scores %s' % scores
    print 'overall score is %.3f' % overallScore

def pruneUniqueNgrams(ngrams):        # ----------------------
    twoOrMore = [n for n in ngrams if ngrams[n] > 1]
    print 'across all docs: %d ngrams; after pruning %d' % (len(ngrams), len(twoOrMore))
    return twoOrMore
```

```
def findAllNgrams(contents):          # ----------------------
    allTrigrams = {}
    merged = ''
    for text in contents:
        for tri in characterTrigrams(text):
            if tri in allTrigrams:
                allTrigrams[tri] += 1
            else:
                allTrigrams[tri] = 1
    return allTrigrams

def targetNumbers(targets, nameInventory):        # ----------------------
    # targets is a list of strings, each a sequence of names
    targetIDs = []
    for target in targets:
      threeNumbers = []
      for name in target.split():
          threeNumbers.append(nameInventory.index(name))
      targetIDs.append(threeNumbers)
    return targetIDs



# main ----------------------------------------------------
import sys, numpy as np

print('......... irStub .........')
contents, names =  parseAlternatingLinesFile('csFaculty.txt')
print 'read in pages for %s' % names
trigramInventory = pruneUniqueNgrams(findAllNgrams(contents))
archive = [computeFeatures(line, trigramInventory) for line in contents]

if len(sys.argv) >= 2 and (sys.argv[1] == 'yesThisReallyIsTheFinalRun'):
    queryFile = 'testQueries.txt'
else:
    queryFile = 'trainingQueries.txt'

queries, targets = parseAlternatingLinesFile(queryFile)
targetIDs = targetNumbers(targets, names)
results = retrieve(queries, trigramInventory, archive)
```

```
modelName = 'silly character trigram model'
scoreAllResults(queries, results, targetIDs, modelName + ' on ' + queryFile)
```