Social Computing
제 6 강
Process Human Language

2014/4/11

Homework – etl로 제출

- 오늘 배운 것을 바탕으로 다음을 수행하세요.

1. TF-IDF
   - Reuter data set중 C50train 디렉토리에 있는 2500개의 기사들 중에 다음 query에 대하여 tf-idf 값이 높은 상위 10개의 기사를 찾아서 각각 리스트를 제출하세요
   - "Hong Kong", "technology", "government", "human rights"

2. Matrix Diagram
   - 지난주 과제로 모은 블로그의 글들을 대상으로 Matrix diagram을 작성하세요
   - 해당 수집 코드를 blog.py 로 저장해서 제출
   - 생성된 JSON화일을 matrix.json로 저장해서 제출
   - matrix..html에서 로드한 결과를 이미지로 저장/캡처해서 matrix.jpg로 제출

3. Reuter 50 50 Data Set
   - Reuter data set중 C50test 디렉토리에 있는 50명의 저자의 첫번째 글을 이용하여 다음을 구하세요.
   - 가장 cosine similarity가 가까운 작가 pair와 가장 먼 작가 pair를 구하세요.

- 4월10일 자정까지 모든 파일을 zip으로 압축해서 etl 을 통해 제출

TF-IDF

- 자료에 포함되어 있는 단어의 중요성에 따라서 연관성을 계산하는 방법
- Term frequency – inverse document frequency
- A numerical ***statistic*** that is intended to reflect how important a word is to a document in a collection or ***corpus***.
- Can be used to ***query*** a ***corpus*** by calculating ***normalized scores*** that express the ***relative importance*** of ***terms*** in documents
- 각 어휘가 무슨 의미인지 잘 알아야 합니다.

Summed TF-IDF for Sample Queries

Document	tf-idf(mr.)	tf-idf(green)	tf-idf(the)	tf-idf(plant)
corpus['a']	0.1053*2.0986 0.2209	0.1053*1.0 0.1053	0.1053*2.099 0.2209	0*1.4055 0
corpus['b']	0*2.0986 0	0.1111*1.0 0.1111	0*2.099 0	0.1111*1.4055 0.1562
corpus['c']	0*2.0986 0	0.0625*1.0 0.0625	0*2.099 0	0.0625*1.4055 0.0878

Query	corpus['a']	corpus['b']	corpus['c']
Green	0.1053	**0.1111**	0.0625
Mr. green	0.2209 + 0.1053 = **0.3262**	0 + 0.1111 = 0.1111	0 + 0.0625 = 0.0625
The green plant	0.2209 + 0.1053 + 0 = **0.3262**	0 + 0.1111 + 0.1562 = 0.2673	0 + 0.0625 + 0.0878 = 0.1503

Cosine Similarity

- Jaccard distance
- Pearson's score (distance)
- Euclidean distance

$$\vec{a} = (0, 3)$$
$$\vec{b} = (4, 0)$$
$$\vec{a} \cdot \vec{b} = 0 * 4 + 3 * 0 = 0$$

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$
$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Process Human Language

- Document summarization
- Generating HTML output
- Entity extraction
- Discovering interactions between entities

Install Numpy

- Windows
  - http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy
  - numpy-MKL-1.8.1.win32-py2.7.exe 설치
- OS X
  - 먼저 XCode를 설치

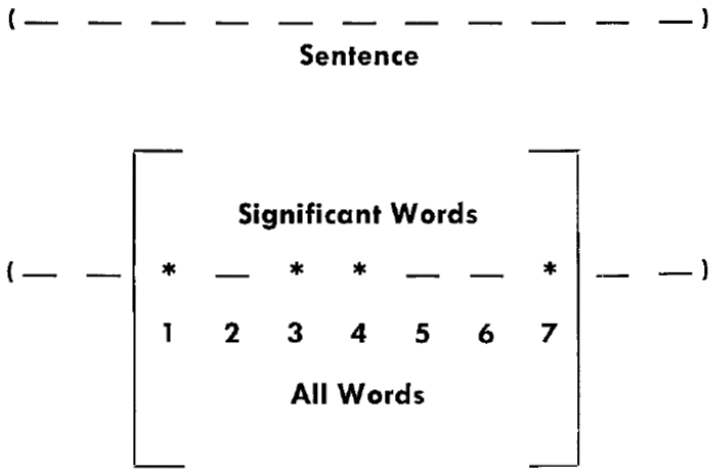
```
export CFLAGS="-arch i386 -arch x86_64"
export FFLAGS="-m32 -m64"
export LDFLAGS="-Wall -undefined dynamic_lookup -bundle -arch i386 -arch x86_64"
export CC=gcc
export CXX="g++ -arch i386 -arch x86_64"

pip install numpy
```

Document Summarization by Luhn's Algorithm

- 긴 글에서 핵심 문장만 추려보자
- "The Automatic Creation of Literature Abstracts" by H.P. Luhn
- IBM Journal 1958

- Score each cluster of tokens using

$$\frac{(\text{significant words in cluster})^2}{\text{total words in cluster}}$$



Portion of sentence bracketed by and including significant words not more than four non-significant words apart. If eligible, the whole sentence is cited.

Figure 2 **Computation of significance factor.** *The square of the number of bracketed significant words (4) divided by the total number of bracketed words (7) = 2.3.*

NLTK를 이용한 문장 분석

- import nltk
- txt = "Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow."

- sentences = nltk.tokenize.sent_tokenize(txt)
- tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]

- normalized_sentences = [s.lower() for s in sentences]
- words = [w.lower() for sentence in normalized_sentences for w in nltk.tokenize.word_tokenize(sentence)]
- fdist = nltk.FreqDist(words)

Text Summarization in Python

```
import json
import nltk
import numpy
import os

AUTHOR_DIR = '/Users/bongwon/Downloads/C50/C50train/AlanCrosby'
#AUTHOR_DIR = 'C:\\temp\\C50\\C50train\\AlanCrosby'

articles = []

for filename in os.listdir(AUTHOR_DIR):
    if not filename.endswith('.txt'):
        continue
    filepath = os.path.join(AUTHOR_DIR, filename)
    file = open(filepath, 'r')
    text = file.read()
    articles += [{'filename':filename, 'text':text}]

N = 100  # Number of words to consider
CLUSTER_THRESHOLD = 5  # Distance between words to consider
TOP_SENTENCES = 5  # Number of sentences to return for a "top n" summary
```

Text Summarization in Python Cont.

```
def _score_sentences(sentences, important_words):
    pass

def summarize(txt):
    return dict(top_n_summary=['AA','BB', 'DD'], mean_scored_summary=['AA','BB','VV'])
```

```
for article in articles:
    #
    article.update(summarize(article['text']))
    #
    print article['filename']
    print '=' * len(article['filename'])
    print
    print 'Top N Summary'
    print '-------------'
    print ' '.join(article['top_n_summary'])
    print
    print 'Mean Scored Summary'
    print '-------------------'
    print ' '.join(article['mean_scored_summary'])
    print
```

Score Sentences Function

```
def _score_sentences(sentences, important_words):
    scores = []
    sentence_idx = -1
    for s in [nltk.tokenize.word_tokenize(s) for s in sentences]:
        sentence_idx += 1
        word_idx = []
        # For each word in the word list...
        for w in important_words:
            try:
                # Compute an index for where any important words occur in the sentence.
                word_idx.append(s.index(w))
            except ValueError, e: # w not in this particular sentence
                pass
        word_idx.sort()
        #
        # It is possible that some sentences may not contain any important words at all.
        if len(word_idx)== 0: continue
        #
        # Using the word index, compute clusters by using a max distance threshold
        # for any two consecutive words.
        clusters = []
        cluster = [word_idx[0]]
        i = 1
        while i < len(word_idx):
            if word_idx[i] - word_idx[i - 1] < CLUSTER_THRESHOLD:
                cluster.append(word_idx[i])
            else:
                clusters.append(cluster[:])
                cluster = [word_idx[i]]
            i += 1
        clusters.append(cluster)
        #
        # Score each cluster. The max score for any given cluster is the score
        # for the sentence.
        max_cluster_score = 0
        for c in clusters:
            significant_words_in_cluster = len(c)
            total_words_in_cluster = c[-1] - c[0] + 1
            score = 1.0 * significant_words_in_cluster * significant_words_in_cluster / total_words_in_cluster
            if score > max_cluster_score:
                max_cluster_score = score
        scores.append((sentence_idx, score))
    return scores
```

Summarize Function

```
def summarize(txt):
    sentences = [s for s in nltk.tokenize.sent_tokenize(txt)]
    normalized_sentences = [s.lower() for s in sentences]
    #
    words = [w.lower() for sentence in normalized_sentences for w in
nltk.tokenize.word_tokenize(sentence)]
    fdist = nltk.FreqDist(words)
     top_n_words = [w[0] for w in fdist.items()
            if w[0] not in nltk.corpus.stopwords.words('english')][:N]
    #
     scored_sentences = _score_sentences(normalized_sentences, top_n_words)
    #
    # Summarization Approach 1:
    # Filter out nonsignificant sentences by using the average score plus a
    # fraction of the std dev as a filter
    #
    avg = numpy.mean([s[1] for s in scored_sentences])
    std = numpy.std([s[1] for s in scored_sentences])
     mean_scored = [(sent_idx, score) for (sent_idx, score) in scored_sentences
                if score > avg + 0.5 * std]
    return dict(top_n_summary=[sentences[idx] for (idx, score) in top_n_scored])
```

Summarize Function

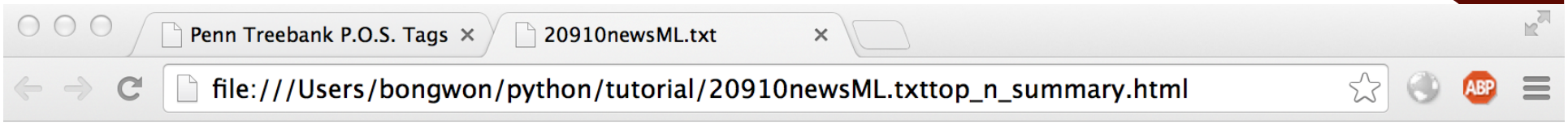
```
def summarize(txt):

    # 위에 코드 그대로

    # Summarization Approach 2:
    # Another approach would be to return only the top N ranked sentences
    #
    top_n_scored = sorted(scored_sentences, key=lambda s: s[1])[-TOP_SENTENCES:]
    top_n_scored = sorted(top_n_scored, key=lambda s: s[0])
    #
    # Decorate the post object with summaries
    #
    return dict(top_n_summary=[sentences[idx] for (idx, score) in top_n_scored],
                mean_scored_summary=[sentences[idx] for (idx, score) in mean_scored])
```

Generating HTML Summarization

```
import os
import json
import nltk
import numpy

HTML_TEMPLATE = """<html>
   <head>
      <title>%s</title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
   </head>
   <body>%s</body>
</html>"""

AUTHOR_DIR = '/Users/bongwon/Downloads/C50/C50train/AlanCrosby'
#AUTHOR_DIR = 'C:\\temp\\C50\\C50train\\AlanCrosby'

articles = []

for filename in os.listdir(AUTHOR_DIR):
   if not filename.endswith('.txt'):
      continue
   filepath = os.path.join(AUTHOR_DIR, filename)
   file = open(filepath, 'r')
   text = file.read()
   articles += [{'filename':filename, 'text':text}]
```

Generating HTML Summarization Cont.

```
for article in articles:
    # Uses previously defined summarize function.
    article.update(summarize(article['text']))
    #
    article['top_n_summary_marked_up'] = '<p>%s</p>' % (article['text'],)
    for s in article['top_n_summary']:
        marked_up = article['top_n_summary_marked_up'].replace(s, '<strong>%s</strong>' % (s, ))
        article['top_n_summary_marked_up'] = marked_up
    filename = article['filename'] + '.topsummary.html'
    f = open(filename, 'w')
    #
    html = HTML_TEMPLATE % (article['filename'], article['top_n_summary_marked_up'],)
    f.write(html.encode('utf-8'))
    f.close()
    print "Data written to", f.name
```



Sweden beat the Czech Republic 3-0 in a World Cup ice hockey game on Thursday, setting up a showdown for European group supremacy against Finland and leaving the reigning world champions searching for answers. The fast-skating Swedes seized control of the game from the opening faceoff, forcing the Czechs to take several early penalties and keeping its vaunted offense in check. Toronto Maple Leaf winger Mats Sundin opened the scoring midway through the first period when he sped around the Czech defense and pulled Dallas Stars' goalie Roman Turek across the crease before sliding the puck between the netminder's legs. **The Swedish National Hockey League (NHL) connection struck again early in the second frame when Washington Capital defenseman Calle Johansson blasted a slapshot from the blue line past a screened Turek with only 57 seconds gone.** Turek, voted best goalie at the world championships for the past two years, was once again screened when a weak shot from Jonas Bergqvist found the bottom corner of the net to close out the scoring for the undefeated Swedes. **The Czechs came into the tournament riding high after capturing the world title in Vienna in May, but have lost the first two games of the tournament and are in danger of crashing out despite having their version of the "Dream Team".** With the addition of Pittsburgh Penguins duo Jaromir Jagr and Petr Nedved, Montreal Canadiens sharpshooter Martin Rucinsky and the core of the world championship team in tact, the Czechs were looking to prove they belonged at the pinnacle of the hockey world. The match at Prague's Sports Hall was to be a homecoming of sorts for the Czechs, their first home game since beating Canada 3-2 in a thrilling world championship final on a last minute goal. But after a demoralising loss to Finland in its tournament opener on Tuesday, and another listless effort on Thursday, Czech fans had had enough, pelting the Czech bench with beer cans near the end of the game. **Even Czech coach Ludek Bukac, a veteran coach with almost 30 years behind the bench, was left searching for an answer to his sputtering offense and the team's lack of dynamism.** "The Swedes played well, you've got to hand it to them. But the fact that we didn't score at home is not a very good showing," he said. Added Jagr: "We aren't that bad, but our performance is not showing it." Bukac, who has often expressed his disdain for bringing in players who have talent but not team spirit refused to comment on whether he would invite the same team back if he could do it all again. **But with nine players on the roster with seven or less of national team games under their belts, the Czech players know they must come together quickly, or face the embarassment of failing to win in what is touted as the true battle for hockey supremacy.** "We've got to concentrate on winning against Germany and moving on to the next round, nothing else," added Robert Reichel, who recently re-signed with the Calgary Flames after playing one year in Germany. The win allows Sweden to keep pace with group leaders Finland, who have also won both their games so far but have a better goal difference. The two teams meet on Sunday in Stockholm while the Czechs must regroup and beat winless Germany in Garmisch-Partenkirchen on Saturday to move into the quarterfinals next week in North America.

Entity Extraction

- 문장의 중심이 무었인가?
- 단순한 단어의 빈도에 따르지 말고 의미를 파악해서 분석을 해보자
- 고유 명사 중심으로 분석

NLTK를 이용한 문장 분석

- import nltk
- txt = "Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow."

- sentences = nltk.tokenize.sent_tokenize(txt)
- tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
- pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]

```
>>> pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]
>>> pos_tagged_tokens
[[('Mr.', 'NNP'), ('Green', 'NNP'), ('killed', 'VBD'), ('Colonel', 'NNP'),
  ('Mustard', 'NNP'), ('in', 'IN'), ('the', 'DT'), ('study', 'NN'),
  ('with', 'IN'), ('the', 'DT'), ('candlestick', 'NN'), ('.', '.')],
 [('Mr.', 'NNP'), ('Green', 'NNP'), ('is', 'VBZ'), ('not', 'RB'),
  ('a', 'DT'), ('very', 'RB'), ('nice', 'JJ'), ('fellow', 'JJ'),
  ('.', '.')]]
```

Part of Speech (POS)

- http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

- NN: Noun, sigular or mass
- NNS: Noun, plural
- NNP: Proper noun, singular
- NNPS: Proper noun, plural

- JJ: Adjective
- RB: Adverb
- RBR: Adverv, comparative
- DT: 관사 the, a
- IN: preposition (전치사) or subordinating conjunction (접속사)

- VB: Verb, base form
- VBD, VBG, VBN, VBP…
-

POS 준비

- txt = "Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow."
- sentences = nltk.tokenize.sent_tokenize(txt)
- tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
- pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]
- pos_tagged_tokens = [token for sent in pos_tagged_tokens for token in sent]
  - 왜 두번 했을까요?

```
>>> pos_tagged_tokens
[('Mr.', 'NNP'), ('Green', 'NNP'), ('killed', 'VBD'), ('Colonel', 'NNP'), ('Must
ard', 'NNP'), ('in', 'IN'), ('the', 'DT'), ('study', 'NN'), ('with', 'IN'), ('th
e', 'DT'), ('candlestick', 'NN'), ('.', '.'), ('Mr.', 'NNP'), ('Green', 'NNP'),
('is', 'VBZ'), ('not', 'RB'), ('a', 'DT'), ('very', 'RB'), ('nice', 'JJ'), ('fel
low', 'JJ'), ('.', '.')]
>>>
```

Entity Extraction in Python

```
import nltk
import json
import os

nltk.download('punkt')
nltk.download('maxent_treebank_pos_tagger')

AUTHOR_DIR = '/Users/bongwon/Downloads/C50/C50train/AlanCrosby'
#AUTHOR_DIR = 'C:\\temp\\C50\\C50train\\AlanCrosby'

articles = []

for filename in os.listdir(AUTHOR_DIR):
    if not filename.endswith('.txt'):
        continue
    filepath = os.path.join(AUTHOR_DIR, filename)
    file = open(filepath, 'r')
    text = file.read()
    articles += [{'filename':filename, 'text':text}]
```

Entity Extraction Cont.

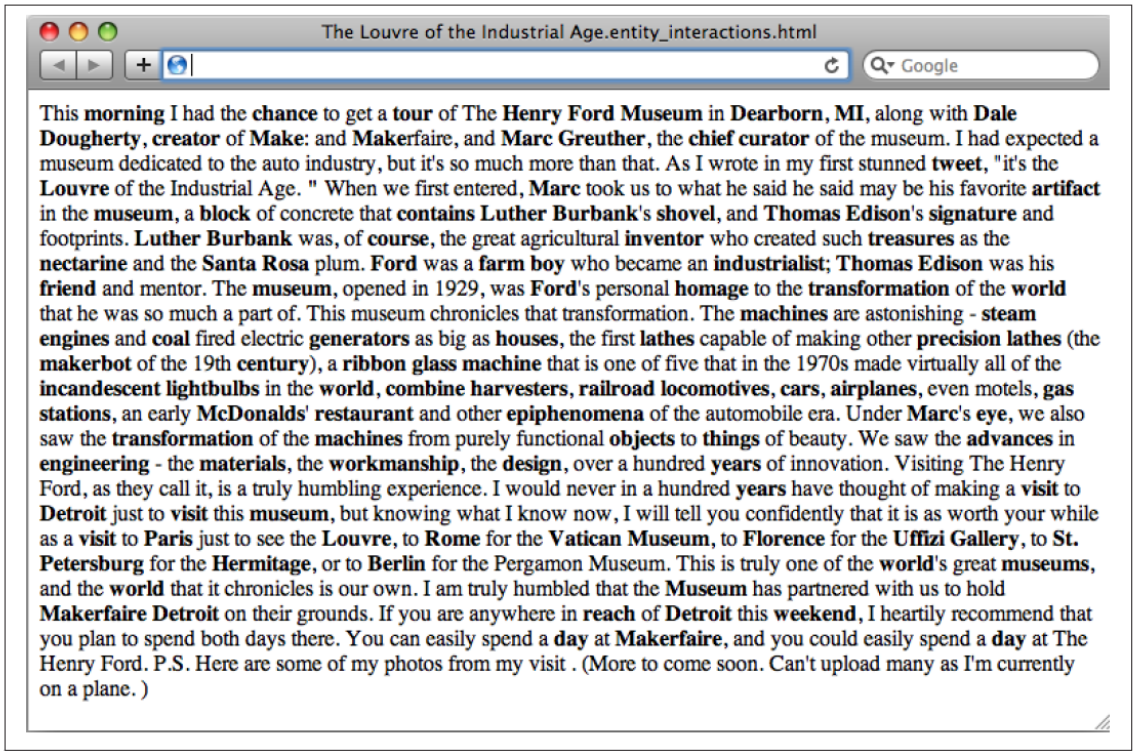
```
for article in articles:
    sentences = nltk.tokenize.sent_tokenize(article['text'])
    tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
    pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]
    pos_tagged_tokens = [token for sent in pos_tagged_tokens for token in sent]
    all_entity_chunks = []
    previous_pos = None
    current_entity_chunk = []
    for (token, pos) in pos_tagged_tokens:
        if pos == previous_pos and pos.startswith('NN'):
            current_entity_chunk.append(token)
        elif pos.startswith('NN'):
            if current_entity_chunk != []:
                all_entity_chunks.append((' '.join(current_entity_chunk), pos))
            current_entity_chunk = [token]
        previous_pos = pos
    print article['filename']
    print '-' * len(article['filename'])
    for (entity, pos) in all_entity_chunks:
        print '\t%s (%s)' % (entity, pos)
    print
```

핵심코드

```
all_entity_chunks = []
previous_pos = None
current_entity_chunk = []
for (token, pos) in pos_tagged_tokens:
    if pos == previous_pos and pos.startswith('NN'):
        current_entity_chunk.append(token)
    elif pos.startswith('NN'):
        if current_entity_chunk != []:
            all_entity_chunks.append((' '.join(current_entity_chunk), pos))
        current_entity_chunk = [token]
    previous_pos = pos
```

Visualization

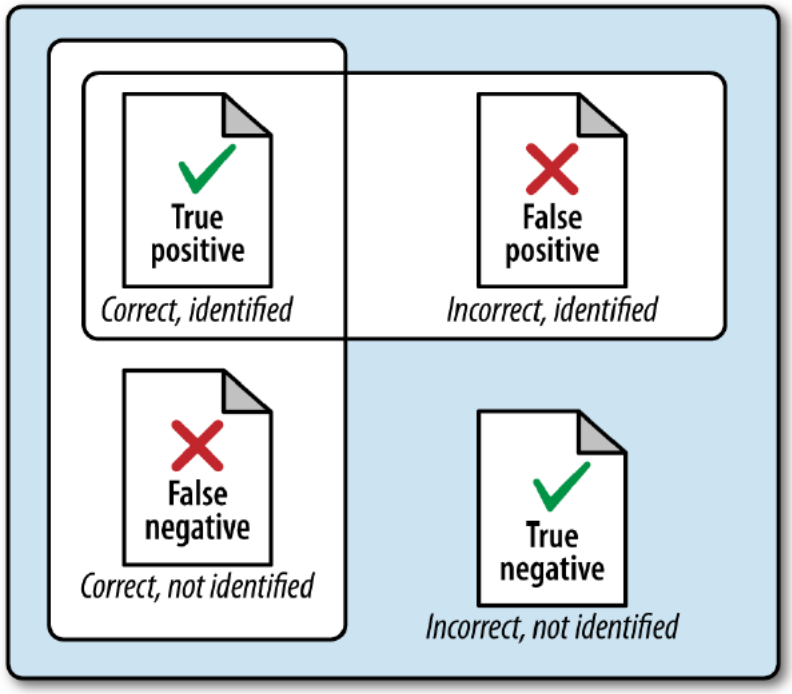
- 교재 pp 215 – 218 참조
- 교재는 blog의 예를 가지고 하지만 우리는 Reuter기사를 가지고 실험함
- 코드를 조금 수정해야 함
- 앞의 예를 보면 어렵지 않게 할 수 있을 것으로 믿습니다!



Precision, Recall, F-Score

- 얼마나 정확한지 평가

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)



- F-score는 precision과 recall의 조화평균
  - 조화평균 H = n / (1/A + 1/B + … 1/Z)
    - 비율의 평균을 구하는데 사용
  - 산술평균 A = (A + B + … Z) / n
  - 40 Km/h 와 60 Km/h로 30분씩 달린 자동차의 평균 속도는?
  - 50 Km/h 인가 48 Km/h인가?

Homework – etl로 제출

- 오늘 배운 것을 바탕으로 다음을 수행하세요.

1. Text Summarization
   - Reuter data set의 C50train 디렉토리에 있는 저자중 임의의 저자를 골라 10개를 summarize 하세요
   - 코드는 summarization.py 로 제출
   - 결과는 XXXXXnewsML.txt.topsummary.html, XXXXXnewsML.txt.meansummary.html 형태로 제출
   - Top n summary방법과 Mean score summay방법중 어떤 방법이 좋았는지 비교 설명한 자료를 summary.pdf로 제출

2. Entity Extraction
   - 위에서 고른 10개의 기사에서 entity를 추출해서 visualization하세요
   - 코드는 entity.py 로 제출
   - 결과는 XXXXXnewsML.txt.entity.html 형태로 제출

3. Precision, Recall, and F-score
   - 위에서 고른 기사들을 직접 읽고 수동으로 entity를 고른후 2번에서 자동으로 추출한 결과와 비교해서 precision, recall, F-score를 구하고 정리해서 analysis.pdf로 제출

- 4월24일 자정까지 모든 파일을 zip으로 압축해서 etl 을 통해 제출

PseudoCode란

- Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.

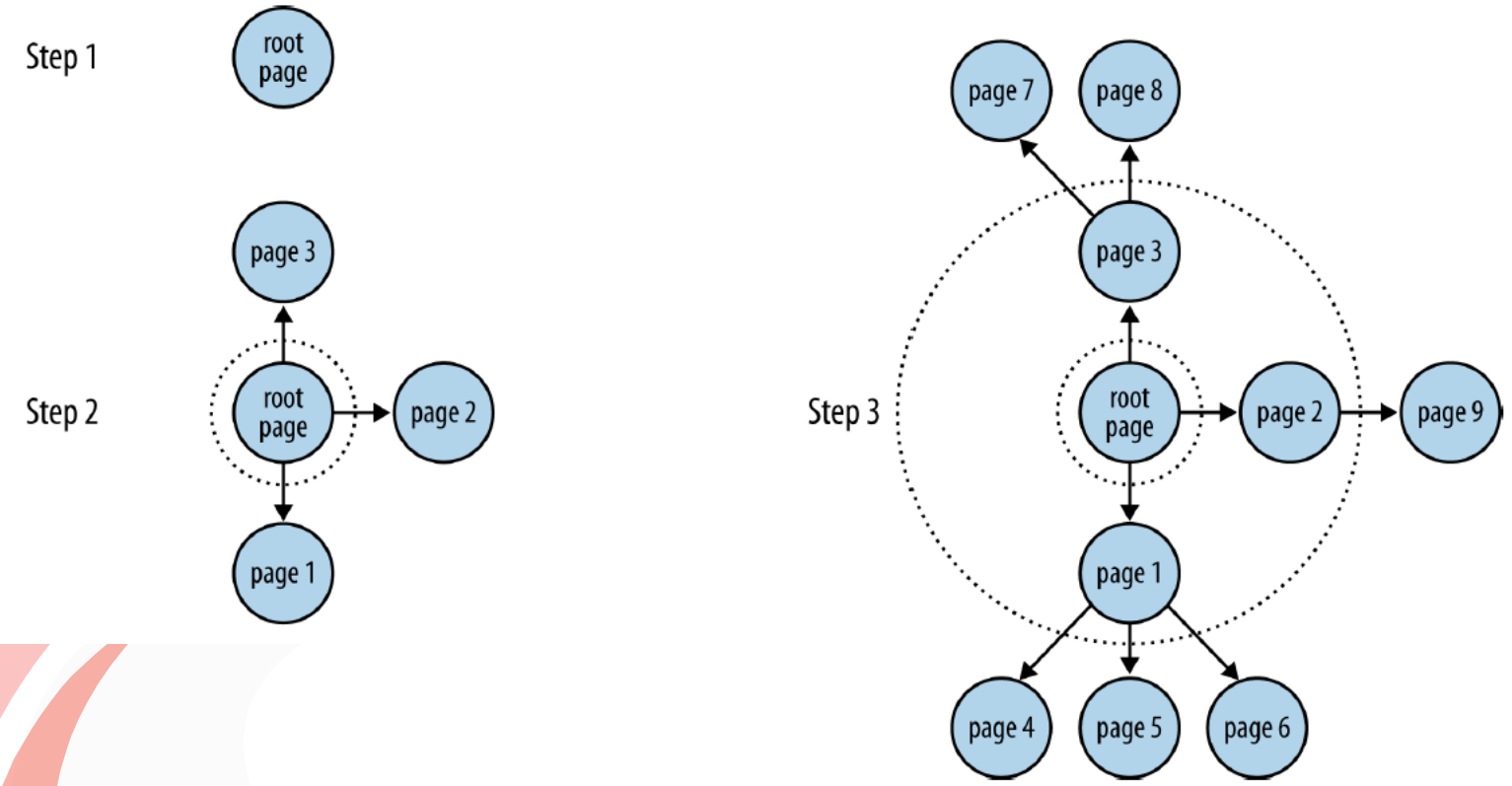
```
Set i to 1.
Repeat this loop:
    If i > n, then exit the loop.
    If A[i] = x, then exit the loop.
    Set i to i + 1.
Return i.
```

The following pseudocode searches the array in the reverse order,

```
Set i to n.
Repeat this loop:
    If i ≤ 0, then exit the loop.
    If A[i] = x, then exit the loop.
    Set i to i − 1.
Return i.
```

http://en.wikipedia.org/wiki/Linear_search

Breadth-first Search



Breadth-first Search

```
Create an empty graph
Create an empty queue to keep track of nodes that need to be processed

Add the starting point to the graph as the root node
Add the root node to a queue for processing

Repeat until some maximum depth is reached or the queue is empty:
  Remove a node from the queue
  For each of the node's neighbors:
    If the neighbor hasn't already been processed:
      Add it to the queue
      Add it to the graph
      Create an edge in the graph that connects the node and its neighbor
```

Hierarchical Clustering

1. Each item is a cluster. (If you have N items, you now have N clusters, each containing just one item.)
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

- 너무 high level description
- 조금 더 디테일이 있으면 좋겠다.

중간고사

- 트위터에서 어느 특정 주제에 대해서 가장 트윗을 많이 한사람을 찾으려 한다. 이를 위해서 수행하여야 하는 task를 10가지 이상의 step으로 상세하게 설명하시요.
- 페이스북에서 ego-centric한 친구관계를 분석하여 grouping을 하려할 경우 수행하여야 하는 task를 10가지 이상의 step으로 상세하게 설명
- K-means clustering code
- Hierarchical clustering code
- 간단한 python for loop 문제

- 구체적인 API이름을 외울 필요는 없다
  - 어디서 찾으면 된다는 것만 알면 된다.
- 코드가 문법적으로 완벽하지 않아도 된다.
  - 무엇을 하려는지 아는 것이 더 중요하다
- 정의나 용어 (e.g. API란?)는 알아야 한다.

중간고사 예

- 트위터에서 어느 특정 주제에 대해서 가장 트윗을 많이 한사람을 찾으려 한다. 이를 수행하기위한 python 프로그램을 작성하는데 필요한 task를 10가지 이상의 step으로 상세하게 설명하시요.
  1. twitter.com/search에서 주제를 반영하는 여러 키워드를 입력하여 검색해보고 키워드가 적절한지 판단
  2. apps.twitter.com 에서 app을 생성하고 API를 이용할때 필요한 key들을 생성
  3. pip install twitter 를 수행하여 twitter python package를 설치
  4. 앞에서 받은 키를 이용하여 다음과 같이 twitter connection을 만듬
     1. auth = twitter.oauch.OAuth(앞의 키들을 입력)
     2. twitter_api = twitter.Twitter(auth=auth)
  5. 다음과 같이 검색을 시도
     1. search_results = twitter_api.search(q='키워드', count=100)
  6. 검색결과를 확인하여 결과로써 어떠한 자료들이 오는지 확인
     1. print json.dumps(search_results)
  7. 트윗의 작성자를 추출하기 위한 key를 확인
     1. statuses = search_results['statuses']
     2. statuses[0]['users']['screen_name']
  8. 작성자를 리스트에 모음
     1. users += [status['users']['screen_name'] for status in statuses]
  9. For loop를 추가하여 더 많은 트윗을 받아올수 있도록 함
     1. (제2강의 24페이지에 있는 코드와 유사)
  10. Counter(users)를 통하여 가장 트윗을 많이 작성한 사람을 출력
      1. from collections import Counter
      2. c = Counter(users)
      3. print c.most_common()[:10]