Assignment #1 (100pts): CSCI 404/573

Out: Jan 10, 2016

Due: Jan 20 midnight (11:59pm), online submission

Submission

1. All files should be submitted to Canvas. Only one of your team members needs to submit on behalf of the team. Indicate clearly in your submission the team members. You can submit multiple times, but please have the same team member resubmit all required files each time.
2. Implementation: Any programming language you are familiar with (Python is strongly recommended for NLP tasks, if you do not have preference).
3. Submit the assn1**.**zip file that contains 3 folders (q1, q2, and q3). **For each question**, submit your programs (if any.) and a written report (**report.** txt, .doc, or .pdf) that presents the results you obtained, summarizes the methods you implemented if any, any additional resources you used, and discuss the results if necessary.
4. **DO NOT** include the given data files in your submission!!

Question 1: Corpus processing: Tokenization and counting with NLTK (60 pts)

NLTK (Natural Langue Toolkit) provides support for a wide variety of text processing tasks: tokenization, stemming, proper name identification, part of speech tagging, and so on. The NLTK website can be found here. NLTK has already been installed on lab machines. Run `python` and then type in `import nltk`, and you are ready to go. You can also choose to install NLTK on your own machine. See next page for references on installation. The NLP book (Natural Language Processing with Python) on NLTK (one of my recommended books for this course) can be found here.

In your report, include how you obtained the results (either via screen shots, or Jupyter Notebook file, or copy the commands you used, or other ways). For this part, your tasks are:

1) Run NLTK sentence tokenizer on the file "sample.txt". Report the number of sentences in that file.
2) Run the NLTK default word tokenizer (**TreebankWordTokenizer**) on the file "nc.txt" where the format is one sentence per line. Report the following numbers on the given corpus (use NLTK built-in functions as much as you can):
    a. Number of tokens.
    b. Number of types (i.e. the vocabulary size).
    c. Type/token ratio (The type/token ratio is defined as the number of types divided by the number of tokens. It roughly tells the lexical diversity/vocabulary richness of the text).
    d. Number of tokens in the vocabulary containing digits [0-9].
    e. Number of tokens in the vocabulary containing punctuation.
    f. Number of tokens in the vocabulary containing both alpha [A-Za-z] and numerics [0-9].
    g. Word count and percentage of the 20 most frequent tokens in the vocabulary (including stopwords and punctuations) (See next page for a small sample output, and see "stopwords.txt" for a list of stop words)

h. Word count and percentage of the 20 most frequent tokens in the vocabulary (excluding stopwords and punctuations).
i. Number and percentage of the singletons, i.e., types that appear only once in the corpus.
j. Frequency and percentage of the 20 most frequent word pairs in the corpus (excluding stopwords and punctuations). (See a small sample output on next page)

3) Run the NLTK **TweetTokenizer** and **WordPunctTokenizer** on the file **"**microblog_tiny.txt" respectively. Discuss on the differences of these two tokenizers based on the results.

A small piece of sample output for part g)

```
20 most common words:
 … …
 of :     7 (1.99%)
 he :     6 (1.70%)
  ; :     6 (1.70%)
'll :     5 (1.42%)
 … …
```

A small sample output for part j):

```
20 most frequent token pairs (excl. stopwords and punctuations)
 … …
('international', 'monetary') :    64 (0.03%)
       ('global', 'economic') :    64 (0.03%)
              ('tax', 'cuts') :    63 (0.03%)
         ('young', 'people') :    63 (0.03%)
      ('financial', 'sector') :    63 (0.03%)
```

References:

1. Installation of python and NLTK: http://berkeley-lit-dh.org/?page_id=51
2. NLTK:
   a. http://www.nltk.org/
   b. the textbook
   c. http://www.nltk.org/api/nltk.tokenize.html
   d. http://textminingonline.com/dive-into-nltk-part-ii-sentence-tokenize-and-word-tokenize
3. Stop words:
   a. https://en.wikipedia.org/wiki/Stop_words
   b. stopwords.txt (see the assignment 1 "release" folder)

Question 2: Minimum Edit Distance (20pts)

The following Python code (distance.py) computes the minimum number of edits: insertions, deletions, or substitutions that can convert an input source string to an input target string. Using the cost of 1, 1 and 2 for insertion, deletion and replacement is traditionally called Levenshtein distance.

```
def distance(target, source, insertcost, deletecost, replacecost):
    n = len(target)+1
    m = len(source)+1
    # set up dist and initialize values
    dist = [ [0 for j in range(m)] for i in range(n) ]
    for i in range(1,n):
        dist[i][0] = dist[i-1][0] + insertcost
    for j in range(1,m):
        dist[0][j] = dist[0][j-1] + deletecost
    # align source and target strings
    for j in range(1,m):
        for i in range(1,n):
            inscost = insertcost + dist[i-1][j]
            delcost = deletecost + dist[i][j-1]
            if (source[j-1] == target[i-1]): add = 0
            else: add = replacecost
            substcost = add + dist[i-1][j-1]
            dist[i][j] = min(inscost, delcost, substcost)
    # return min edit distance
    return dist[n-1][m-1]

if __name__=="__main__":
    from sys import argv
    if len(argv) > 2:
        print "levenshtein distance =", distance(argv[1], argv[2], 1, 1, 2)
```

Let's assume we save this program to the file `distance.py`, then:

```
$ python distance.py gamble gumbo
levenshtein distance = 5
```

Your task is to produce the following visual display of the best (minimum distance) alignment:

```
$ python view_distance.py gamble gumbo
levenshtein distance = 5
g a m b l e
|   | |
g u m b _ o
```

```
$ python view_distance.py "recognize speech" "wreck a nice beach"
levenshtein distance = 14
_ r e c _ _ o g n i z e   s p e e c h
  | | |       | |   | |     |   | |
w r e c k   a   n i c e   _ b e a c h

$ python view_distance.py execution intention
levenshtein distance = 8
_ e x e c u t i o n
      |     | | | |
i n t e _ n t i o n
```

The 1st line of the visual display shows the target word and the 3rd line shows the source word. An insertion in the target word is represented as an underscore in the 3rd line aligned with the inserted letter in the 1st line. Deletion from the source word is represented as an underscore '_' in the 1st line aligned with the corresponding deleted character in the source on the 3rd line. Finally, if a letter is unchanged between target and source then a vertical bar (the pipe symbol '|') is printed aligned with the letter in the 2nd line.

You can produce this visual alignment using two different methods:

- Memorize which of the different options: insert, delete or substitute was taken as the entries in the table are computed; or
- Trace back your steps in the table starting from the final distance score by comparing the scores from the predecessor of each table entry and picking the minimum each time.

There can be many different alignments that have exactly the same minimum edit distance. Therefore, for the above examples producing a visual display with a different alignment but which has the same edit distance is also correct.

2) Print out the valid alignments with the same minimum edit distance. You should print out the first 100 alignments or N alignments, where N comes from a command line argument -n. This is essential because the number of possible alignments is exponential in the size of the input strings.

Question 3: Language Model and Noisy Channel (20pts)

Read through the slides “Spell Correction” on Canvas, write an essay describing step by step how the spelling correction is perform by using the example of correcting “acress” on the slides. Make sure to show your understanding to the process. Clarity and correctness is required.