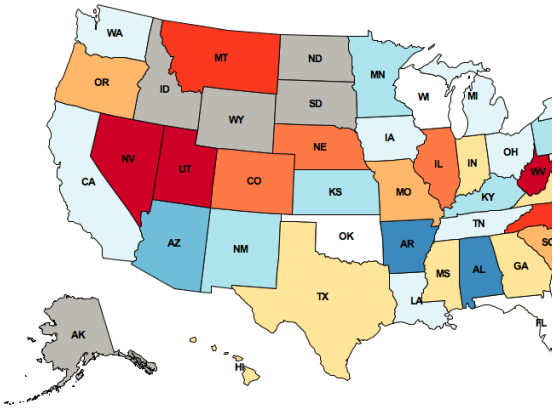
Programação 1 (IF968)

Search this site

Projeto

Tendências do Twitter



O que as pessoas twitam?
Desenhe os sentimentos delas em
um mapa para descobrir
tendências.

Introdução

Neste projeto, você desenvolverá uma visualização geográfica de dados de twitter nos EUA. Você precisará usar dicionários, listas e técnicas de abstração de dados para criar um programa modular.

O mapa mostrado acima descreve como as pessoas em estados diferentes sentem sobre Texas. Esta imagem é gerada através de:

1. Coleta de publicações do Twitter (tweets) que foram marcadas com locais geográficos e filtragem para aquelas que contêm o termo de consulta "texas";
2. Atribuição de um sentimento (positivo ou negativo) para cada tweet, com

base em todas as palavras que aparecem nesse tweet;
3. Agregação de tweets pelo estado americano com o centro geográfico mais próximo, e finalmente
4. Pintura de cada estado de acordo com o sentimento agregado de seus tweets. Vermelho significa sentimento positivo; Azul significa negativo.

Os detalhes de como realizar cada uma dessas etapas são descritos a seguir. No final deste projeto, você será capaz de mapear o sentimento de qualquer palavra ou frase. Existem dois arquivos zip alternativos relacionados a este projeto:

- a versão completa, que contém todo o código inicial e todos os dados (**aviso: 81 MB**).
- uma versão pequena que contém todo o código inicial, mas apenas um subconjunto pequeno dos dados. Você pode fazer o projeto todo usando apenas essa versão e, no final, adicionar mais dados usando a versão completa.

O projeto usa diversos arquivos, mas todas as suas modificações serão aplicadas ao primeiro deles:

trends.py	Uma implementação inicial e incompleta do arquivo principal do projeto.
geo.py	Posições geográficas, equações de projeção 2-D e funções de distância geográfica.
maps.py	Funções para desenhar mapas.
data.py	Funções para carregar dados do Twitter a partir de arquivos.

graphics.py	Uma biblioteca gráfica simples em Python.
ucb.py	Funções utilitárias.

O diretório de dados (data) contém todos os arquivos de dados necessários para o projeto. Os arquivos zip acima contém este diretório, mas se você baixar os arquivos de projeto individuais acima (também aparecem na seção de anexos desta página), você precisará baixar o diretório de dados também.

Logística

Este é um projeto de duas semanas. Você vai trabalhar em uma equipe de até duas pessoas e você pode completar todos os problemas junto com seu parceiro.

Comece cedo! Sinta-se livre para pedir ajuda cedo e muitas vezes. A equipe do curso está aqui para ajudá-lo, mas não podemos ajudar a todos uma hora antes do prazo.

O código do arquivo **trends.py** deverá estar disponível em um repositório no Github (http://www.github.com). Deve ser enviado um **email** para o professor com o assunto **PROJETO IF968** até as **23h59 do dia 21/11/16** informando o endereço do repositório. Programas não entregues no horário serão penalizados com a perda de pontos, à taxa de 2,0 pontos perdidos para cada 24h de atraso.

As apresentações dos projetos serão realizadas **impreterivelmente no dia 21/11/2016**, no horário da aula. **Todos os membros** da equipe **precisam estar presentes**, sob pena de receber nota zero.

Trabalhos copiados de colegas ou da internet, seja trechos ou totalidade, serão prontamente anulados (todos os envolvidos).

Fase 1: Os Sentimentos nos Tweets

Nesta fase, você criará um tipo abstrato de dados para tweets, dividirá o texto dos tweets em palavras e calculará a quantidade de sentimentos negativos ou positivos em um tweet.

Tweets

Primeiro, você implementará um tipo de dados abstrato para Tweets. A função `make_tweet` é definida no início de `trends.py`. `make_tweet` devolve um dicionário que contém as seguintes entradas:

```
text:      um string. o texto de um tweet,
time:      um objeto datetime, quando o t
latitude:  um número de ponto flutuante,
longitude: um número de ponto flutuante,
```

Problema 1 . Implemente as funções `tweet_words` e `tweet_time`. Use a função `extract_words` para listar as palavras no texto de um tweet.

Problema 2. Implemente a função `tweet_location`, que devolve uma `position` (posição). Posições são outro tipo de dados, definido no início do arquivo `geo.py`. Certifique-se de que entende como manipular posições; elas desempenham um papel importante neste projeto.

Quando você completar os problemas 1 e 2, o doctest para `make_tweet` deve passar.

```
python3 trends.py -t make_tweet
```

Problema 3. Melhore a função `extract_words` da seguinte maneira: suponha que uma palavra é qualquer sub-string consecutivo de `text` que consiste apenas de letras ASCII. A string `ascii_letters` do módulo `string` contém todas as letras do conjunto de caracteres ASCII. A função `extract_words` deve listar todas as palavras em ordem e nada mais. Para importar o módulo `string` em seu programa, inclua a linha

```
import string
```

no início dele. Feito isso, você pode acessar `ascii_letters` usando `string.ascii_letters`.

Quando terminar este problema, o doctest para `extract_words` deve passar.

```
python3 trends.py -t extract_words
```

Problema 4. Implemente o tipo de dados abstrato `sentiment`, que representa um valor de sentimento que pode ou não existir. A função `make_sentiment` recebe ou um valor numérico entre -1 e 1 (sendo -1 muito negativo e 1 muito positivo) ou `None` para indicar que o valor de sentimento não existe. Implemente também as funções `has_sentiment` e `sentiment_value`. Você pode usar qualquer representação que quiser para sentimentos (tupla, dicionário, etc.), mas o resto do programa não deve depender dessa representação.

Quando terminar este problema, devem passar os doctests para `make_sentiment` e `get_word_sentiment`. Você também pode chamar a função `print_sentiment` para imprimir os valores de sentimento de todas as palavras que têm um sentimento associado em uma linha de texto.

```
python3 trends.py -t make_sentiment
python3 trends.py -t get_word_sentiment
python3 trends.py -p computer science is m
python3 trends.py -p life without lambda: a
```

Problema 5. Implemente a função `analyze_tweet_sentiment`, que recebe um tweet como parâmetro (do tipo de dados abstrato definido no início desta seção) e devolve um `sentiment`. Leia os docstrings para `get_word_sentiment` e `analyze_tweet_sentiment` p entender como as duas funções interagem. **Lembrando**: *sua implementação não deve depender da representação de* `sentiment`!

Quando terminar este problema, os doctests para `analyze_tweet_sentiment` devem passar.

```
python3 trends.py -t analyze_tweet_sentime
```

Fase 2: A Geometria de Mapas

Posições

Usaremos o tipo de dados abstrato `position` para representar as posições geográficas de latitude-longitude na Terra. Associadas a esse tipo abstrato de dados, há três funções definidas em `geo.py`, `make_position`, `latitude` e `longitude`.

Nesta fase, você irá escrever duas funções que, em conjunto, determinam os centros dos estados dos Estados Unidos. A forma de um estado é representada como uma lista de polígonos. Alguns estados (por exemplo, Havaí) consistem em polígonos múltiplos, mas a maioria dos estados (por exemplo Colorado) consistem em apenas um polígono (ainda representado como uma lista de comprimento 1).

Problema 6. Implemente a função `find_centroid` , que recebe um polígono como parâmetro e devolve três valores: as coordenadas de seu centróide e sua área. O polígono de entrada é representado como uma lista de valores do tipo abstrato `position`, que são os vértices consecutivos de seu perímetro. O primeiro vértice é sempre idêntico ao último.

O centróide de uma forma bidimensional é seu centro de equilíbrio, definido como a interseção de todas as linhas retas que dividem uniformemente a forma em metades de área igual. A função `find_centroid` deve devolver o centróide e a área de um polígono individual.

A fórmula para calcular o centróide de um polígono está na Wikipédia. A fórmula parte do pressuposto de que os vértices são consecutivos (no sentido horário ou anti-horário, ambos dão a mesma resposta), uma propriedade que você pode supor que sempre é válida para a entrada.

A área de um polígono nunca é negativa. Dependendo de como você calcular a área, talvez seja necessário usar a função `abs` para retornar um número não negativo. Essa função recebe um número como parâmetro e devolve seu valor absoluto:

```
>>> abs(10)
10
>>>abs(-42)
42
```

Quando você completar esse problema, o doctest para `find_centroid` deve passar.

python3 trends.py -t find_centroid

Problem 7. Implemente `find_center`, que recebe uma forma representada por uma lista de polígonos e devolve uma posição, seu centróide. Note que a diferença fundamental
entre `find_center` e `find_centroid` é que a segunda funciona com apenas um polígono enquanto a segunda devolve o centróide de uma forma representada por múltiplos polígonos.

O centróide de uma coleção de polígonos pode ser computado por decomposição geométrica. O centróide de uma forma é a média ponderada dos centróides de seus polígonos componentes, ponderados por sua área.

Quando você completar esse problema, o doctest para find_center deve passar.

python3 trends.py -t find_center

Uma vez que tenha terminado, a função `draw_centered_map` desenhará os `10` estados mais próximos de um dado estado, incluindo o próprio estado.

python3 trends.py -d CA

Fase 3: O Humor de uma Nação

Estados

A variável `us_states` referencia um dicionário que contém a forma de cada estado dos EUA, com o seu código postal de duas letras. Você pode usar as chaves deste dicionário para iterar sobre todos os estados dos Estados Unidos.

Nesta fase, você vai escrever funções para determinar o estado de onde um tweet vem, agrupar tweets por estado e calcular o sentimento positivo ou negativo médio em todos os tweets associados a um estado.

Problema 8. Implemente a função `find_closest_state`, que devolve um código postal de duas letras do estado que é o mais próximo ao local de um tweet. Use a função `geo_distance` (disponível em `geo.py`) para calcular a distância mais curta em milhas entre duas posições.

Quando terminar este problema, os doctests para `find_closest_state` devem passar.

```
python3 trends.py -t find_closest_state
```

Problema 9.
Implemente `group_tweets_by_state`, que recebe uma seqüência de tweets como parâmetro e devolve um dicionário. As chaves do dicionário devolvido são nomes de estado (códigos postais de duas letras) e os valores são listas de tweets que aparecem mais próximos do centro desse estado do que de qualquer outro.

Quando você completar esse problema, o doctests para `group_tweets_by_state` deve passar.

```
python3 trends.py -t
group_tweets_by_state
```

Problema 10. Como exercício, implemente `most_talkative_state`, que devolve o estado que possui mais tweets que contém um dado termo.

Quando terminar esse exercício, os doctests de `most_talkative_state` devem passar.

python3 trends.py -t most_talkative_state

Problem 11.
Implemente `average_sentiments`. Esta função recebe o dicionário devolvido por `group_tweets_by_state` e também devolve um dicionário. As chaves do dicionário devolvido são os nomes de estados (códigos postais de duas letras) e os valores são valores de sentimento médio para todos os tweets nesse estado.

Se um estado não tiver tweets com valores de sentimento, deixe-o totalmente fora do dicionário retornado. Evite de incluir um estado sem sentimento usando um valor de sentimento zero. Zero representa o sentimento neutro e não o sentimento desconhecido. Os estados com sentimento desconhecido aparecerão cinzentos enquanto os estados com sentimentos neutros aparecerão brancos.

Você agora deve ser capaz de desenhar mapas que são coloridos pelo sentimento correspondente aos tweets que contêm um determinado termo.

python3 trends.py -m sandwich
python3 trends.py -m obama
python3 trends.py -m texas
python3 trends.py -m my life

Se você baixou a versão pequena do projeto, só será capaz de mapear esses quatro termos. Se você quiser mapear qualquer termo, você precisará da versão completa do projeto, mais especificamente, do arquivo `all_tweets.txt`, que deve ficar no diretório `data` do seu projeto.

Fase 4: Entrando na Quarta Dimensão

Nesta última seção, você levará em conta a quarta dimensão: o tempo. Cada tweet tem um objeto `datetime`, que representa o instante em que o tweet foi postado.

A função `draw_map_by_hour` fornecida visualiza os tweets que foram postados durante cada hora do dia. Por exemplo, sweets mencionando "sandwich" aparecem mais positivos às 22:00: lanche tarde da noite!

Problema 12.
Implementar `group_tweets_by_hour`, que recebe uma seqüência de tweets e devolve um dicionário. As chaves do dicionário devolvido são os inteiros de 0 a 23, representando as 24 horas do dia. O valor para cada chave é uma lista dos tweets que foram postados durante essa hora.

Para começar, leia a documentação on-line sobre objetos `datetime`. Quando você terminar, você poderá visualizar a forma como sentimentos sobre sanduíches mudam com o passar do tempo.

Python3 trends.py -b sandwich

Agradecimentos: Aditi Muralidharan desenvolveu este projeto com John DeNero. Hamilton Nguyen o estendeu. Fernando Castor traduziu parcialmente a especificação para português.

DATA.PY (3K)	FERNANDO J...	V.1	
GEO.PY (3K)	FERNANDO J...	V.1	
GRAPHICS.PY ('	FERNANDO J...	V.1	
MAPS.PY (3K)	FERNANDO J...	V.1	
TRENDS.PY (13	FERNANDO J...	V.1	
TRENDS_SM...	FERNANDO J...	V.1	
UCB.PY (3K)	FERNANDO J...	V.1	

ADICIONAR ARQUIVOS

Comentários

Claudio Victor Rosas Pacheco

Adicionar um comentário