Online topic-sentiment mining

Finn Årup Nielsen

***Abstract*—We describe a lightweight webservice that performs online topic mining with sentiment analyze using standard components of Python. It can analyze a small corpus on a few hundred small documents in a few hundred milliseconds.**

I. INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

II. METHODS

The webservice does not use a web framework, rather just relying on CGI with an Apache webserver.

For topic and sentiment mining we use `re`, `numpy` and `scipy` packages. We use `scipy.sparse` module for sparse matrix rather than the full matrices in standard NumPy as the bag-of-words matrices will usually be quite sparse.

For topic mining we used non-negative matrix factorization in a form of an algorithm suggested by Lee and Seung [1]. Rather than relying on its implementation in the `sklearn.decomposition` module we implemented the algorithm from the bottom, see Figure 1. For speed the algorithms uses a default of only 50 iterations.

For sentiment analysis we used a word list approach relying on the AFINN word list with 2477 words labeled for valence [2]. Although there are convenient machine learning classifiers in `nltk` and `scikits` packages we could train for classification of sentiment we did not have an appropriate data set for training the classifier.

The script runs in Python 2 as some of the libraries we used were not readily available in their Python 3 versions on the system we developed and ran on.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

The web service is a single CGI script and the listing begins at page 3.

III. RESULTS

Figure 2 shows a screenshot of the webservice with data copy-and-pasted from the Wikipedia article on Lundbeck.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
for n in range(0, iterations):
    H = np.multiply(H, (W.T * M) / (W.T * W * H + 0.001))
    W = np.multiply(W, (M * H.T) / (W * (H * H.T) + 0.001))
```

Fig. 1. Central part of the NMF algorithm implemented in three lines of Python code.

A. Code checking

We used pylint to check our coding quality.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

B. Testing

We wrote a script that called the webscript and checked the returned result.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

C. Profiling

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

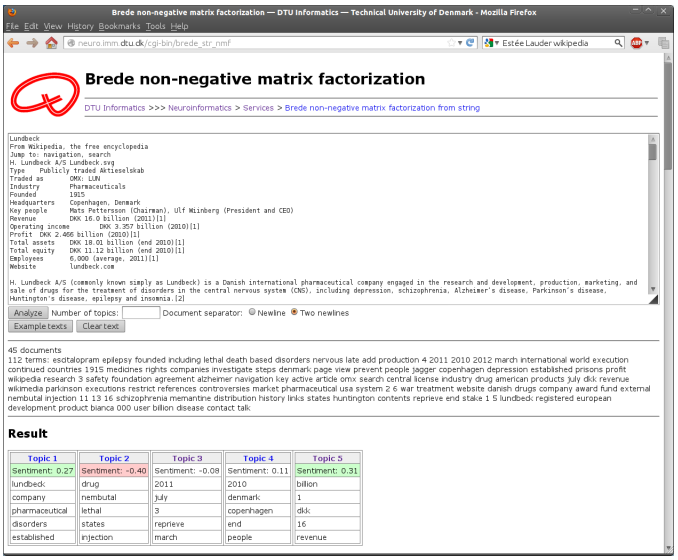

Fig. 2. Web service screenshot with text from Wikipedia article “Lundbeck”.

IV. Discussion

With the sentiment wordlist the sentiment analysis will only work for English texts.

There are numerous issue with the implementation: The HTML templates are not appropriately separated from the code, the functions `components2html_*` have redundant code, a variable called `wordsfreq` could be implemented as a `collections.Counter`, etc. Obviously unit testing could have be done if the code was well structured into modules, e.g., with the `nose` module. The simple word tokenization with the regular expression “`\w+`” faults for some words, e.g., “won’t” is tokenized to the two tokens “won” and “t” and as “won” is positive in the AFINN word list a positive bias is introduced.

There are different sparse matrix representations in scipy.sparse (csr, csc and lil). A proper profiling may have shown that the “lil” format used to set up the matrix is not the most efficient in the iterative algorithm and the documentation suggests “once a matrix has been constructed, convert to CSR or CSC format for fast arithmetic and matrix vector operations”. Furthermore, it is not clear how well sparse matrices works in conjunction with dense matrices: Should `W` and `H` in the `nmf` function have been made sparse too?

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

V. Conclusion

We implemented a fast on-the-fly topic-sentiment mining web service suitable for small corpora.

References

[1] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, Massachusetts: MIT Press, 2001, pp. 556–562. [Online]. Available: http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf

[2] F. Å. Nielsen, “A new ANEW: evaluation of a word list for sentiment analysis in microblogs,” in *Proceedings of the ESWC2011 Workshop on ’Making Sense of Microposts’: Big things come in small packages*, ser. CEUR Workshop Proceedings, M. Rowe, M. Stankovic, A.-S. Dadzie, and M. Hardey, Eds., vol. 718, May 2011, pp. 93–98. [Online]. Available: http://ceur-ws.org/Vol-718/paper_16.pdf

Appendix A
Code listings

Listings


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#    Web service for non-negative matrix factorization of a list of strings for topic mining
#
# $Id: brede_str_nmf,v 1.20 2011/11/28 16:38:38 fn Exp $

__version__ = '$Revision: 1.20 $'
__author__  = '$Author: fn $'
__all__     = [ 'brede_str_nmf' ]

import time
time_start = time.time()

filebase = "/var/local/www/"

from cgi    import FieldStorage, escape
import math
import numpy as np
import os
from re     import compile, findall, split, sub, DOTALL, UNICODE
from string import strip, lower
from scipy  import sparse
from urllib import urlopen, urlencode
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

import scipy
scipy_version = map(int, scipy.__version__.split('.'))

warnings = []

pattern_word = compile(r"""\w+""", UNICODE)

# Load sentiment word list
filenameAFINN = filebase + "/AFINN-111.txt"
try:
    afinn = dict(map(lambda (w, s): (unicode(w, 'utf-8'), int(s)), [
          ws.strip().split('\t') for ws in open(filenameAFINN) ]))
except:
    warnings.append('Could not read sentiment word list')
    afinn = {}


def stopwords():
    """
    Return the stopwords as a list.
    If the stopword file is not available then return an empty list.
    """
    filename = filebase + "/stop_english1.txt"
    try:
        words = [ unicode(line.strip(), 'utf-8') for line in open(filename).readlines() ]
    except:
        warnings.append('Could not read stopword list')
        words = []
    return words


def example_texts():
    """
    Example texts
    """
    return u"""Denmark, officially the Kingdom of Denmark together with Greenland and the Faroe Islands, is a Scandinavian country in N
Sweden, officially the Kingdom of Sweden, is a Nordic country on the Scandinavian Peninsula in Northern Europe. Sweden has land borders
Germany, officially the Federal Republic of Germany, is a country in Western Europe. It is bordered to the north by the North Sea, Denm
Australia, officially the Commonwealth of Australia, is a country in the Southern Hemisphere comprising the mainland of the Australian
The Republic of Botswana is a sub Saharan country located in Southern Africa. The citizens are referred to as "Batswana". Formerly the
Oman, officially the Sultanate of Oman, is an Arab country in southwest Asia on the southeast coast of the Arabian Peninsula.
Lego (trademarked in capitals as LEGO) is a line of construction toys manufactured by the Lego Group, a privately held company based in
Nestlé S.A. is the largest consumer packaged goods company in the world, founded and headquartered in Vevey, Switzerland.
Tetra Pak is a multinational food processing and packaging company of Swedish origin.
Arla Foods is a Swedish-Danish cooperative based in Århus, Denmark, and the largest producer of dairy products in Scandinavia.
H. Lundbeck A/S is a Danish international pharmaceutical company engaged in the research and development, production, marketing, and sa
Novo Nordisk manufactures and markets pharmaceutical products and services.
The Carlsberg Group is a Danish brewing company founded in 1847 by J. C. Jacobsen after the name of his son Carl. The headquarters are


def str2html(s):
```

```
80      """
81      Expects a Unicode string and converts it to UTF-8 and escapes it.
82      """
83      return escape(s.encode('utf-8'))
84
85
86  def parseform(inform):
87      outform = {'data': '',
88                 'components': 1,
89                 'format': 'default',
90                 }
91      try:
92          outform['data'] = unicode(inform.getvalue("data", default=""), 'utf-8')
93      except:
94          warnings.append('Could not convert input text to Unicode')
95      u_format = inform.getvalue("format", default='default')
96      if u_format in ['default', 'script', 'scriptfm']:
97          outform['format'] = u_format
98
99      u_components = inform.getvalue("components", default=None)
100     components = 1
101     if u_components and u_components.isdigit():
102         components = int(u_components.strip())
103         if components < 1:
104             components = 1
105         outform['components'] = components
106     return outform
107
108
109 def header():
110     """
111     Returns a string with HTML header
112     """
113     return """Content-type: text/html; charset=utf-8
114
115 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
116 <html>
117   <head>
118     <title>Brede non-negative matrix factorization &mdash; DTU Informatics &mdash; Technical University of Denmark</title>
119
120     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
121     <meta http-equiv="Content-language" content="en">
122
123     <meta name="description" content="Non-negative matrix factorization">
124     <meta name="rating"      content="General">
125
126     <link rel="author" href="http://www.imm.dtu.dk/~fn/">
127     <link rel="home" href="http://neuro.imm.dtu.dk">
128
129     <link rel="stylesheet" href="/css/brede.css" type="text/css">
130
131     <script type="text/javascript">
132       function example_texts() {
133         var text = "%s";
134         return text;
135       }
136     </script>
137   </head>
138
139   <body>
140     <table class="noborder">
141       <tr>
142         <td width="135">
143           <img width="135" height="135" src="/icon/bredewiki.png">
144         <td width="100%%">
145           <h1>Brede non-negative matrix factorization</h1>
146           <hr>
147           <a href="http://www.imm.dtu.dk/English.aspx">DTU Informatics</a> &gt;&gt;&gt;
148           <a href="http://neuro.imm.dtu.dk/home.html">Neuroinformatics</a> &gt;
149           <a href="http://neuro.imm.dtu.dk/services/services.html">Services</a> &gt; <a href="?">Brede non-negative matrix factorizatio
150           <hr>
151     </table>
152     """ % (sub(r'\n', r'\\n', sub(r'"', r'\\"', str2html(example_texts()))), )
153
154
155 def footer():
156     return """
157     <hr>
158     <a href="http://www.imm.dtu.dk/~fn/">Finn &Aring;rup Nielsen</a>,
159     <a href="http://www.imm.dtu.dk/English.aspx">DTU Informatics</a>
160     &mdash; %.3f seconds
161   <body>
162 </html>""" % (time.time() - time_start)
163
164
165 def inputform(data="", components=None):
```

```
    if not components:
        components = 5
    return """
    <form name="inputform" action="/cgi-bin/brede_str_nmf" method="POST">
      <textarea style="width:100%%;" name="data" rows="20" type="text">%s</textarea>
      <br>
      <input type="submit" value="Analyze">
      Number of topics: <input type="text" name="components", value="%d" size="5">
      <input type="button" onClick="data.value = example_texts(); components.value = 2;" value="Example texts">
      <input type="button" onClick="data.value = '';" value="Clear text">
    </form>

    """ % (str2html(data), components)


def description():
    return """
    <hr>
    <h2>Description</h2>
    This web service will perform topic mining with sentiment analysis,
    see <a href="?format=script">the script</a>
    (<a href="?format=scriptfm">formatted</a>).
    <h3>Procedure</h3>
    <ol>
    <li>Type in texts: one in each line.
    <li>Set the number of topics.
    <li>Press "Analyze" and wait.
    </ol>

    <h3>The results</h3>
    The results with the texts will be grouped in topics.
    The value in parentheses after a word or a text is the "load"
    of the word or the text on the topic.

    <h3>Details</h3>
    <ol>
    <li>The topic mining is performed with <a
    href="http://en.wikipedia.org/wiki/Non-negative_matrix_factorization">non-negative
    matrix factorization</a>.
    <li>The sentiment analysis via the <a href="http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010">AFINN</a> word list
        The sentiment of a topic is found by summing sentiment of the individual        texts weighted by the number of texts in the to
        The sentiment of each individual text is found by summing the
        sentiment strength of each word weighted by the number of words.
        The weighting is by the square root.

    <li>A word list excludes common English words ("stopwords")
    <li>The analysis will only work on up to a few hundreds short texts.
    <li>The results may change each time you run the algorithm.
        This is due to random initializations and the issues in the
        factorization algorithms.
    <li>The value shown after each topic, each word and each document is the load       telling how important they are.
    <li>Texts and words are assigned exclusively to one topic
        even if some of the texts are load partially on two or more
        topics.
    <li>If the load of some texts or words are zero then they show up in
        the "not assigned" category.
    <li>The example texts are taken as the first (few) line(s) in
        Wikipedia articles about companies and countries.
        If the topic mining works well it should separate country and company
        texts.
    </ol>

    <h3>References</h3>
    <ol>
    <li><a href="http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3661/pdf/imm3661.pdf">Mining the posterior cingulate: Segregation
    <li><a href="http://ceur-ws.org/Vol-718/paper_16.pdf">A new ANEW:
    Evaluation of a word list for sentiment analysis in
    microblogs</a>, Finn Årup Nielsen,
    Proceedings of the ESWC2011 Workshop on 'Making Sense of
    Microposts': Big things come in small packages, May 2011.
    </ol>
    </div>
"""



def texts2matrix(texts):
    """
    Convert a list of strings to a bag-of-words matrix.
    A sparse scipy matrix and identified terms are returned.
    A stopword list is applied if available.
    """
    wordlists = [ map(lower, pattern_word.findall(text)) for text in texts ]
    wordsfreq = {}
    for wordlist in wordlists:
```

```
252             for word in set(wordlist):
253                 wordsfreq[word] = wordsfreq.get(word, 0) + 1
254         # Remove single instance words
255         terms = [ word for word in wordsfreq if wordsfreq[word] > 1 ]
256         terms = list(set(terms).difference(stopwords()))
257         if terms:
258             M = sparse.lil_matrix((len(texts), len(terms)))
259             for n in range(len(texts)):
260                 for m in range(len(terms)):
261                     M[n,m] = wordlists[n].count(terms[m])
262         else:
263             M = None
264         return (M, terms)
265 
266 
267     def wta((W, H)):
268         """
269         Winner-take-all function
270         """
271         return (np.asarray(map(lambda col: map(lambda e: e*int(e==col.max()), col), np.asarray(W))),
272                 np.asarray(map(lambda row: map(lambda e: e*int(e==row.max()), row), np.asarray(H).T)).T)
273 
274 
275     def adjustwh((W, H)):
276         """
277         Change the scaling of W and H, and record them according to scaling
278         """
279         wsum = np.asarray(W.sum(axis=0)).flatten()
280         hsum = np.asarray(H.sum(axis=1)).flatten()
281         whsum = wsum * hsum
282         whsumsqrt = np.sqrt(whsum)
283         I = np.argsort(-whsum)
284         Dw = np.mat(np.diag(whsumsqrt[I]/wsum[I]))
285         Dh = np.mat(np.diag(whsumsqrt[I]/hsum[I]))
286         return (W[:,I]*Dw, Dh*H[I,:])
287 
288 
289     def nmf(M, components=None, iterations=50):
290         """
291         Non-negative matrix factorization
292         """
293         if not components:
294             components = np.ceil(np.sqrt(float(min(M.shape))/2))
295         W = np.mat(np.random.rand(M.shape[0], components))
296         H = np.mat(np.random.rand(components, M.shape[1]))
297         if scipy_version[0] == 0 and scipy_version[1] < 7 and components > 1:
298             for n in range(0, iterations):
299                 H = np.multiply(H, (W.T * M).todense() / (W.T * W * H + 0.001))
300                 W = np.multiply(W, (M * H.T).todense() / (W * (H * H.T) + 0.001))
301         else:
302             for n in range(0, iterations):
303                 H = np.multiply(H, (W.T * M) / (W.T * W * H + 0.001))
304                 W = np.multiply(W, (M * H.T) / (W * (H * H.T) + 0.001))
305         return (W, H)
306 
307 
308     def text2sentiment(text):
309         """
310         Text as string as input.
311         A float for valence is returned. Positive for positive valence.
312         """
313         words = pattern_word.findall(text.lower())
314         sentiments = map(lambda word: afinn.get(word, 0), words)
315         if sentiments:
316             sentiment = float(sum(sentiments))/math.sqrt(len(sentiments) + np.finfo(float).eps)
317         else:
318             sentiment = 0
319         return sentiment
320 
321 
322     def components2html(W, H, texts, terms):
323         """
324         Turn the text and terms into HTML
325         """
326         weight = np.asarray(W.sum(axis=0)).flatten()
327         iw = np.argsort(-W, axis=0)
328         ih = np.argsort(-H, axis=1)
329         # Iterate over all topics
330         for n in range(W.shape[1]):
331             print("""<h3><a name="topic_%d">Topic %d (%.2f)</a></h3>""" % (n+1, n+1, weight[n]))
332             for m in range(len(terms)):
333                 if H[n,ih[n,m]] > 0:
334                     print("%s (%.2f) " % (str2html(terms[ih[n,m]]), H[n,ih[n,m]]))
335                 else:
336                     break
337             print("<ol>")
```

```
338             sentiments = []
339             for m in range(len(texts)):
340                 if W[iw[m,n],n] > 0:
341                     t = texts[iw[m,n]]
342                     if afinn:
343                         sentiments.append(text2sentiment(t))
344                     print("<li>%s (%.2f)" % (str2html(t), W[iw[m,n],n]))
345             print("</ol>")
346             if afinn:
347                 sentiment = sum(sentiments)/math.sqrt(len(sentiments) + np.finfo(float).eps)
348                 print("Sentiment: %.2f" % sentiment)
349 
350         # Write out documents not assigned to a topic
351         print("""<h3><a name="topic_0">Not assigned</a></h3>""")
352         for m in range(len(terms)):
353             if not any(H[:,m]):
354                 print("%s" % (str2html(terms[m])))
355         print("<ol>")
356         sentiments = []
357         for m in range(len(texts)):
358             if not any(W[m,:]):
359                 t = texts[m]
360                 if afinn:
361                     sentiments.append(text2sentiment(t))
362                 print("<li>%s" % (str2html(t)))
363         print("</ol>")
364         if afinn:
365             sentiment = sum(sentiments)/math.sqrt(len(sentiments) + np.finfo(float).eps)
366             print("Sentiment: %.2f" % sentiment)
367 
368 
369 
370 
371 # Get CGI field value
372 form = parseform(FieldStorage())
373 
374 if form['format'] == 'default':
375 
376     print(header())
377 
378     lines = split(r"\n", form['data'])
379     texts = ""
380     if lines:
381         texts = lines
382 
383     text = "\n".join(texts)
384     print(inputform(data=text, components=form['components']))
385 
386     if texts:
387         matrix, terms = texts2matrix(texts)
388         print("""
389       <hr>
390       %d lines<br>
391       %d terms: %s""" % (len(lines), len(terms), str2html(" ".join(terms))))
392         print("""<hr>
393       <h2>Results</h2>""")
394         if matrix:
395 
396             left_matrix, right_matrix = wta(adjustwh(nmf(matrix, components=form['components'])))
397             components2html(left_matrix, right_matrix, texts, terms)
398         else:
399             print("(no results)")
400 
401     if warnings:
402         print("""
403     <hr>
404     <h2>Warnings</h2>""")
405         for warning in warnings:
406             print("""<div class="warning">%s</div><br>""" % str2html(warning))
407 
408     print(description())
409     print(footer())
410 
411 elif form['format'] == 'script':
412 
413     print("Content-type: text/plain; charset=utf-8\n")
414     filename_script = os.environ.get("SCRIPT_FILENAME", "").strip()
415     s = open(filename_script).read()
416     print(s)
417 
418 elif form['format'] == 'scriptfm':
419 
420     print(header())
421     print("<pre>")
422     filename_script = os.environ.get("SCRIPT_FILENAME", "").strip()
423     s = open(filename_script).read()
```

```
424         print(str2html(s))
425         print("</pre>")
426         print(description())
427         print(footer())
```

APPENDIX B
AUTOMATIC GENERATION OF DOCUMENTATION

Demontration using epydoc:

```
epydoc --pdf -o /home/fnielsen/tmp/epydoc/ --name RBBase wikipedia/api.py
```

This example does not use `brede_str_nmf` but another more well-documented module called `api.py` that are used to download material from Wikipedia.

RBBase

API Documentation

November 19, 2012

Contents

1 Module rbbase.wikipedia.api

RBBase

rbbase/wikipedia/api.py

1.1 Class Api

object
└ **rbbase.wikipedia.api.Api**

Interface to *http://en.wikipedia.org/w/api.php* for getting individual pages, revisions, and differences between revisions.

```
>>> import rbbase.wikipedia.api
>>> api = rbbase.wikipedia.api.Api()
>>> page = api.get_page_by_title("Lundbeck")
```

1.1.1 Methods

___init___(*self*)

Setup of URL to English Wikipedia API and name of user agent.

Overrides: object.___init___

get__page__by__title(*self*, *title*)

Query the Wikipedia API by title.

Exceptions are not handled.

Parameters

`title:` Title of Wikipedia page as unicode

Return Value

`dict` Dictionary with fields. The text in in the 'text' field

```
>>> api = rbbase.wikipedia.api.Api()
>>> page = api.get_page_by_title("Lundbeck")
```

get_revision_by_id(*self*, *revid*)

Query the Wikipedia API for a revision.

Exceptions are not handled.

Parameters

`revid`: Revision identifier as string or number

Return Value

`dict` Dictionary with fields. The text in in the 'text' field

```
>>> api = rbbase.wikipedia.api.Api()
>>> revision = api.get_revision_by_id(233192)
>>> revision["user"]
u'RoseParks'
```

get_diff(*self*, *revid*)

Query Wikipedia API for diff

NOTE: 'changes' missing

Parameters

`revid`: Revision identifier as string or number

Return Value

Dictionary with change information

```
>>> import rbbase.wikipedia.api
>>> api = rbbase.wikipedia.api.Api()
>>> diff = api.get_diff(508885541)
```

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

1.1.2 Properties

Name	Description
Inherited from object	
___class___	

Index