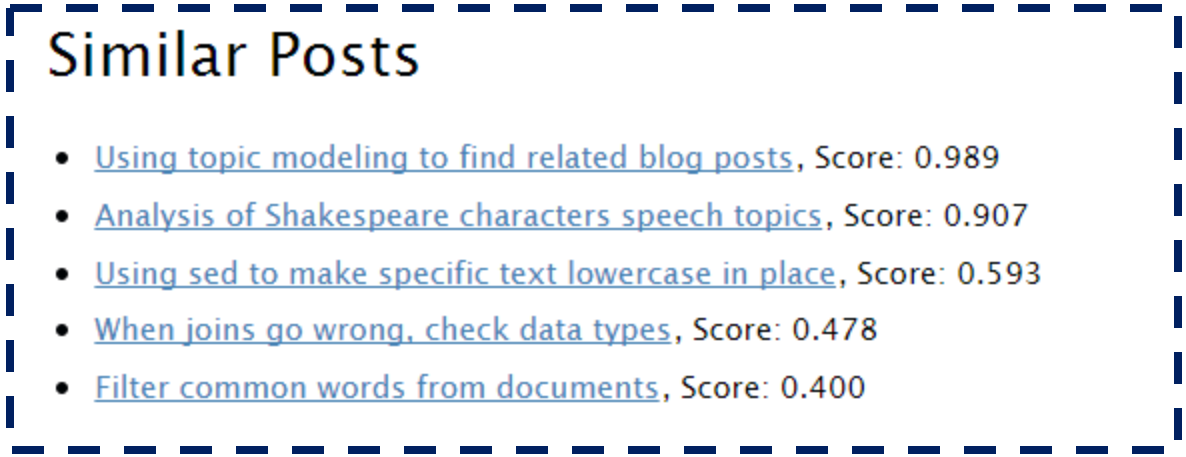
Using topic modeling to add similar posts links to a Pelican powered blog

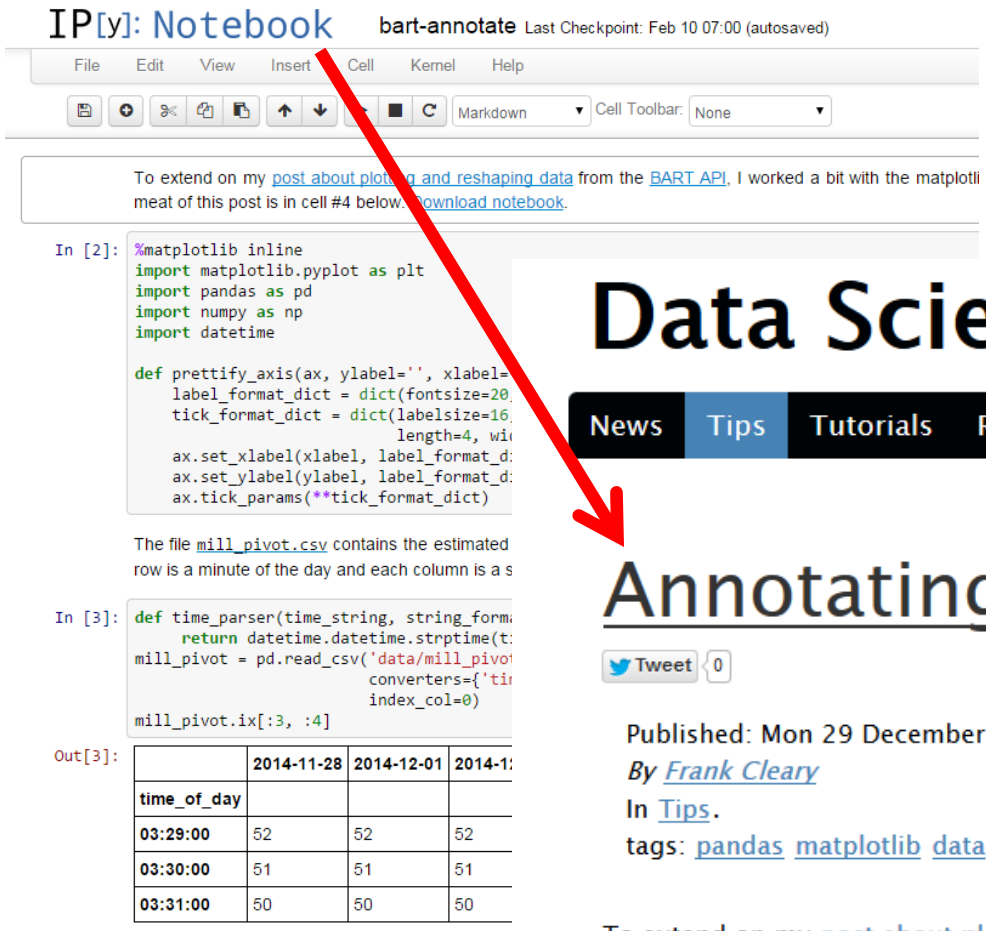


Frank Cleary

www.datasciencebytes.com (hobby)

Member Technical Staff at Index (day job)

Pelican – site generation in Python



(github.com/danielfrg/pelican-ipynb)

Data Science Bytes

News | Tips | Tutorials | Recommended Books | Recommended Videos | About

Annotating matplotlib plots

Tweet 0

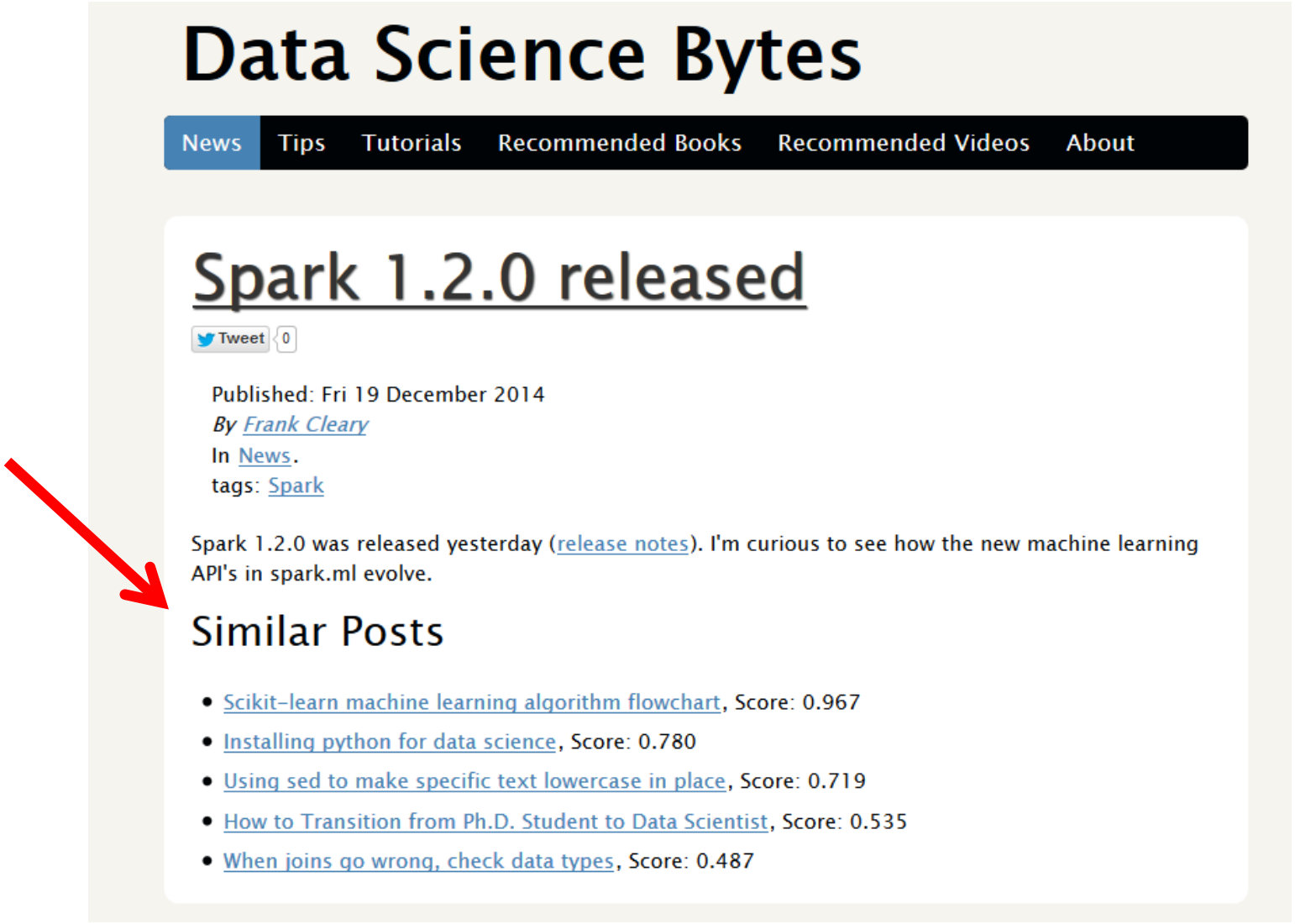
Published: Mon 29 December 2014
By Frank Cleary
In Tips.
tags: pandas matplotlib data python

To extend on my post about plotting and reshaping data from the BART API, I worked a bit with the matplotlib annotion interface to add text and arrows to a plot. The meat of this post is in cell #4 below. Download notebook.

In [2]:

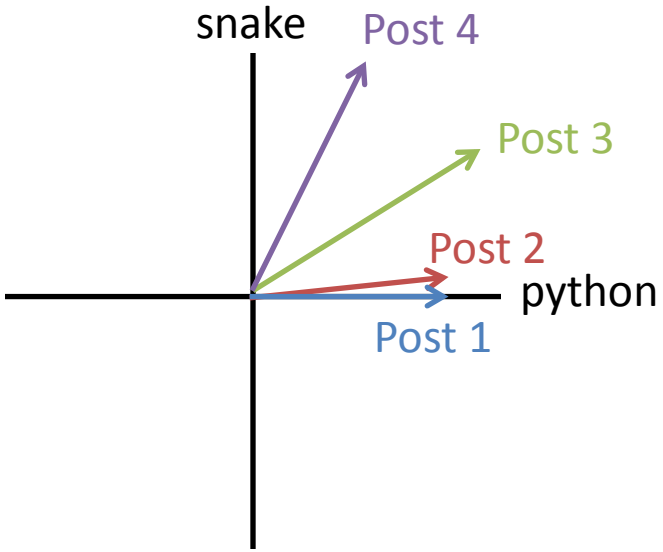
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
```

Adding similar posts



Finding similar posts

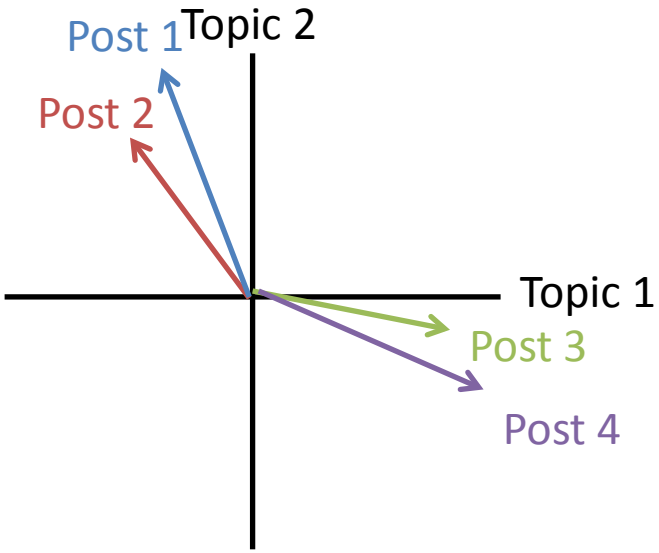
	Word Count			
Word	**Post 1**	**Post 2**	**Post 3**	**Post 4**
python	10	5	12	6
snake	0	1	5	10
iterator	5	4	0	0
data	8	2	0	1
…	…	…	…	…



- Here I've already selected words that differentiate topics. Topic models can do this automatically.

Topic model

	Contribution from Topic			
Topic	**Post 1**	**Post 2**	**Post 3**	**Post 4**
python, snake, scale	-.2	-.3	.5	.6
iterator, data, import	.8	.5	-.2	-.4
sweet, awesome, bug	.2	.2	.2	.2
…	…	…	…	…

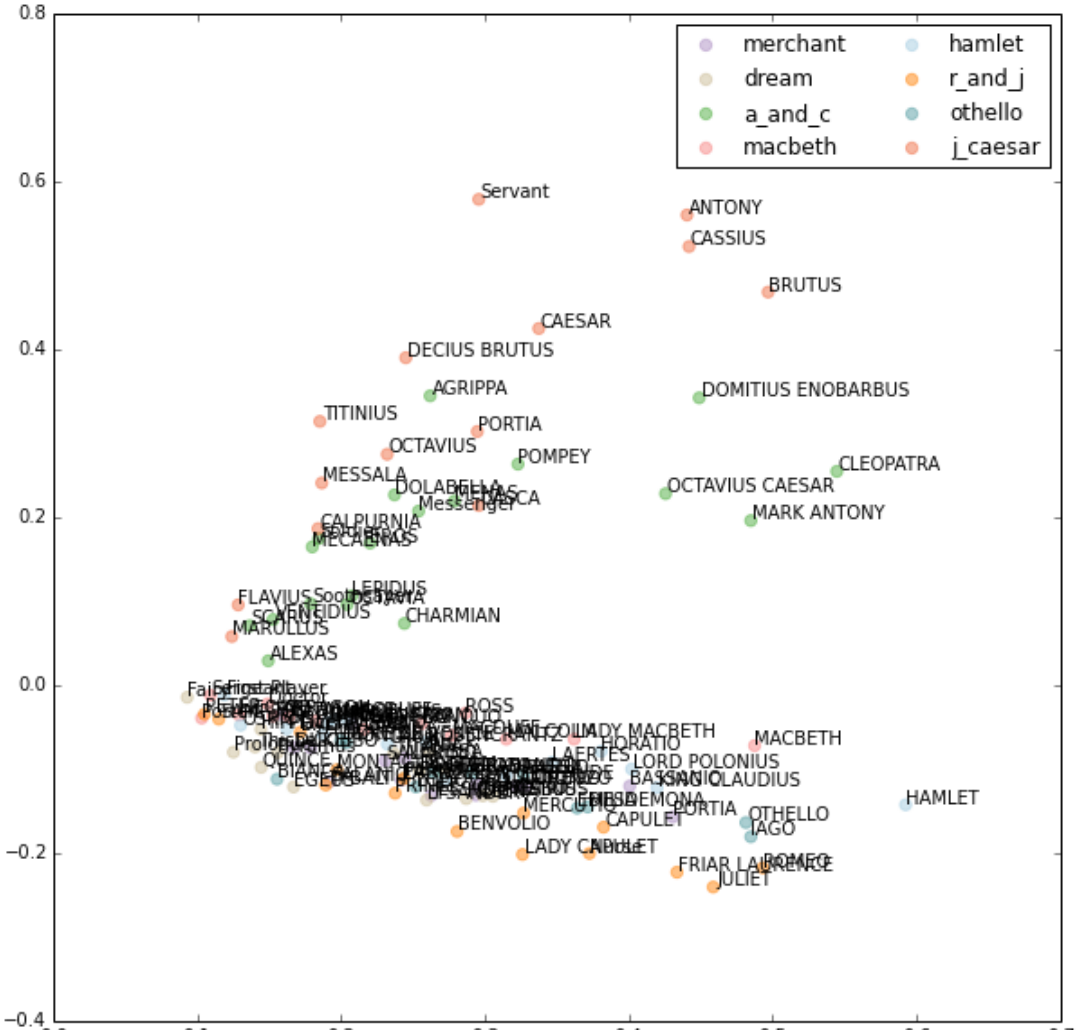


Gensim – topic modeling for humans

Here are two topics of Shakespeare character's speech.

www.datasciencebytes.com/bytes/2014/12/31/analysis-of-shakespeare-character-speech-topics/

```
x topic:
('0.192', u'caesar'),
('0.125', u'lord'),
('0.121', u'antony'),
('0.112', u'brutus'),
('0.106', u'thou'),
('0.105', u'romeo'),
('0.093', u'cassio'),
('0.091', u'love'),
('0.084', u'thee'),
('0.078', u'madam'),
 ...
y topic:
('0.513', u'caesar'),
('0.378', u'brutus'),
('0.286', u'antony'),
('0.192', u'cassius'),
('-0.151', u'romeo'),
('0.139', u'rome'),
('-0.108', u'cassio'),
('0.090', u'octavius'),
('0.081', u'lepidus'),
('-0.073', u'tybalt')
...
```



Implementation

- Hook into Pelican at site-generation time:

```
def register():
    """Entry point for ArticleGenerator from pelican"""
    signals.article_generator_finalized.connect(add_related_posts)
```

- Process articles:

```
def add_related_posts(generator, default_max_related_posts=5):
    max_posts = generator.settings.get("MAX_RELATED_POSTS",
                                       default_max_related_posts)
    similarity_scores = recommend_articles(generator.articles, max_posts)
    articles_by_path = {art.source_path: art for art in generator.articles}
    for article in generator.articles:
        related_posts = similarity_scores[article.source_path]
        article.related_posts = []
        article.score = {}
        for source_path, similarity in related_posts:
            related_post = articles_by_path[source_path]
            article.score[related_post.source_path] = similarity
            article.related_posts.append(related_post)
```

Implementation

- Compute similar articles:

```
def recommend_articles(articles, max_posts,
                       tokenizer=nltk.RegexpTokenizer(r'\w+')):
    article_texts = [BeautifulSoup(article.content).body.get_text().lower()
                     for article in articles]
    documents = [tokenizer.tokenize(text) for text in article_texts]
    index = generate_similarity_index(documents, max_posts)
    similarity_scores = defaultdict(list)
    for article, sims in zip(articles, index):
        for id, score in sims[1:]:
            similarity_scores[article.source_path].append(
                (articles[id].source_path, score)
            )
    return similarity_scores
```

```
def generate_similarity_index(documents, max_posts, model=models.LsiModel):
    dictionary = corpora.Dictionary(documents)
    filter_dictionary(dictionary)
    corpus = [dictionary.doc2bow(doc) for doc in documents]
    tfidf = models.tfidfmodel.TfidfModel(corpus=corpus)
    topic_model = model(tfidf[corpus], id2word=dictionary, num_topics=5)
    for topic in topic_model.print_topics():
        print '\n' + topic
    return similarities.MatrixSimilarity(topic_model[tfidf[corpus]],
                                          num_best=max_posts + 1)
```

Implementation

- Add to article template:

```
{% if article.related_posts %}
<h1>Related Posts</h1>
  <ul>
    {% for related_post in article.related_posts %}
      <li><a href="{{ SITEURL }}/{{ related_post.url }}">
                  {{ related_post.title }}
          </a>,
      Score: {{ '%0.3f' % article.score.get(related_post.source_path) }}</li>
    {% endfor %}
  </ul>
{% endif %}
```

Thanks!

- Pelican: docs.getpelican.com
- Gensim: radimrehurek.com/gensim/
- IPython notebook plugin: github.com/danielfrg/pelican-ipynb
- Pelican tag-based related posts: github.com/getpelican/pelican-plugins/tree/master/related_posts
- SF Python Meetup

- See More: www.datasciencebytes.com (site source: github.com/frankcleary/data-science-bytes)
- Careers at Index: index.com/careers