Document Gathering

July 9, 2015

A document shall be defined as the text body of either a comment or post. Here, we'll gather many subreddit's corpora of documents, and the 2-gram transitions of those documents as laid out in comment threads.

For each subreddit, we can simply traverse each post's comment tree in DFS order so to accomplish gathering the bodies and transitions at the same time. We'll store the document's text bodies in a list, and the transitions as a list of tuples containg the indices of their corresponding documents.

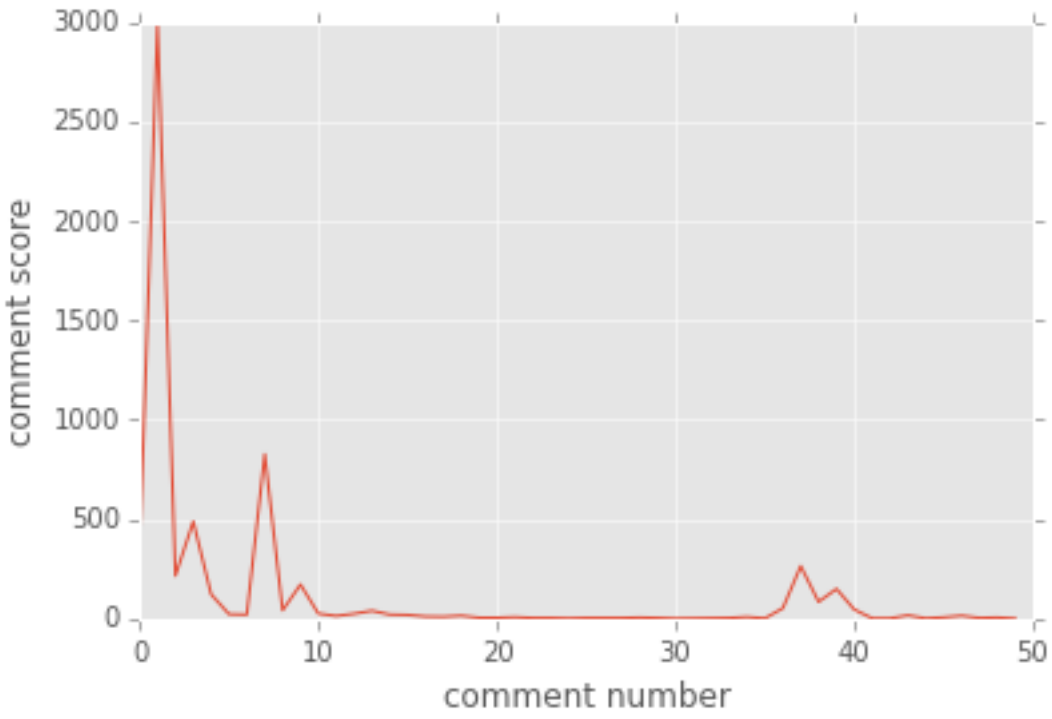
I've come back from the future, and realized that I'll also need to define a generator for a document's comments—since it only fetches several at a time. More precisely, it will fetch a list of comments, with a `MoreComments praw` object at the end. But, what if I arbitrarily want more comments!?

```
In [1]: def all_comments(comments, limit=100):
            for c in comments:
                if isinstance(c, praw.objects.Comment):
                    yield c
                    limit -= 1
                else:
                    for c in all_comments(c.comments(), limit=limit):
                        yield c
                        limit -=1
                if limit <= 0:
                    raise StopIteration
```

And just to play around, I wonder how the comments are sorted:

```
In [2]: import praw
        import matplotlib.pyplot as plt
        %matplotlib inline
        plt.style.use('ggplot')
```

```
In [3]: r = praw.Reddit('test')
        comments = next(r.get_subreddit('science').get_top_from_all(limit=1)).comments
        plt.plot([c.score for c in all_comments(comments, limit=50)])
        plt.xlabel('comment number')
        plt.ylabel('comment score')
        plt.show()
```



Cool, so the comments seem to come in roughly score-sorted order. This way getting the comments from the top posts of all time, for a given subreddit, will be roughly cut from the same principle.

To start, we'll need an interface to login to Reddit. The API wrapper `praw` suits this need perfectly. Now, to define some functions to gather the comments, and transitions.

```
In [4]: from pyprind import ProgPercent
```

```
In [5]: def sub_gather(subreddit, width=5, depth=5):
            docs = []
            trans = []
            r = praw.Reddit('comment LDA')
            sub = r.get_subreddit(subreddit)
            prog = ProgPercent(width)
            for p in sub.get_top_from_all(limit=width):
                docs.append(p.title)
                parent_index = len(docs) - 1    # index of the just added document
                for c in all_comments(p.comments, width):
                    comment_recur(c, parent_index, width, depth, docs, trans)
                prog.update()
            return docs, trans
```

```
In [6]: def comment_recur(comment, parent_index, width, depth, docs, trans):
            if comment.body == '[deleted]' or depth <= 0:
                return
            docs.append(comment.body)
            trans.append((parent_index, len(docs) - 1))
            parent_index = len(docs) - 1    # index of th just added document
            for child in all_comments(comment.replies, limit=width):
                comment_recur(child, parent_index, width, depth - 1, docs, trans)
```

In [ ]:
```
import pickle
```

In [ ]:
```
subreddit = 'AskReddit'
width, depth = 10, 3
docs, trans = sub_gather(subreddit, width=width, depth=depth)
with open('{}[{}-{}].p'.format(subreddit), 'wb') as f:
    pickle.dump((subreddit, docs, trans), f)
```

```
[ 40 %] elapsed[sec]: 329.335 | ETA[sec]: 494.002
```

In [ ]: