Build a Python script that implements the following functionality:

```
   $ python hyperlinks.py --url http://someurl.com
--limit 1000 --out links.json
```

This will traverse the Web as a linked graph from the starting `--url` finding all outgoing links (`<a>` tag): it will store each outgoing link for the URL, and then repeat the process for each of them, until `--limit` URLs will have been traversed.

The output will be a JSON file with the following information:

```
{
    'someurl.com': {
        'incoming': [
            'further.com', ... list of URLs that link to
someurl.com ...
        ],
        'outgoing': [
            'another.com', ... list of URLs linked from
someurl.com ...
        ]
     },
     'another.com': {
        'incoming': [
            'someurl.com', ... list of URLs that link to
another.com ...
        ],
        'outgoing': [
            'further.com', ... list of URLs linked from
another.com ...
        ]
     },
     'further.com': {
        'incoming': [
            'another.com', ... list of URLs that link to
another.com ...
        ],
        'outgoing': [
            'someurl.com', ... list of URLs linked from
```

```
another.com ...
        ]
    }
}
```

(this would be the output with `--limit 3`).

For efficiency, you *may* limit the number of `outgoing` URLs stored to only be up to `--limit`, but this is optional.

If the subset of the Web traversed is sufficiently small, the `incoming` lists may all very well contain just the one link you just traversed (and the first one be empty): this is fine and should be expected.

It is also possible that we have selected an isolated subgraph of the Web, and that the traversal may terminate before we reach `--limit` URLs: this is fine too (although somewhat unexpected).

In the absence of an `--out` option, please output the JSON to stdout (bonus point for pretty printing); both `--url` and `--limit` are required.

NOTES:

- Please assume (naively) that all URLs point to HTML pages
- Bonus point for non-naive implementation
- Extra points for smart use of RegEx (this will also greatly simplify your code)
- Extra bonus points if you can implement a `--dbout` flag that causes the data to be stored in a MongoDB collection (tip: using pymongo saves a lot of pain)
- We expect to see a few unit tests implemented, ideally using the `mock` framework, if you really want to impress us.