# OpenSherlock PubMed Workflow

latest: 20150801
GitHub Repo: https://github.com/KnowledgeGarden/Carrot2PubMedStudyEngine



## Background

There exists an inspiration for a *never-ending learning system* [1], which drives the thinking here. The game, overall, is to learn the biomedical domain through reading combinations of domain text books, and PubMed documents, in both abstract and full document form.

The approach taken for reading PubMed abstracts has been to use the Carrot$^2$ document clustering search engine [2]. In general, interesting terms are hand-fed into the Carrot$^2$ Workbench, then followed with a series of additional terms. In general, those additional terms are predicates typically encountered (Appendix A) or concepts (Appendix B). An example query series (in part) is:

AMPK
AMPK causes
AMPK is caused by
AMPK inflammation
AMPK dementia
…

Till now, that process has been conducted by hand. It consisted of these steps:

1. Run Carrot$^2$ queries by hand
2. Parse the XML query results files to create a list of PubMed documents (by ID) to fetch
3. Fetch each PubMed abstract as an XML file.

4. Convert each PubMed abstract to a JSON document
5. Decorate each JSON document with additional features which include cluster names, and query terms used to find it

Step 3 is problematic in this sense: bulk fetching from the PubMed servers is subject to terms of use, which means that fetching can take place only between the hours of 9pm and 5am, Eastern Time, and all day over weekends. Fetching cannot occur faster than one file every couple of seconds. Thus, the process just described is very slow and labor intensive. That leads us to current thinking about this process.

# Current Thinking

As it turns out, the Carrot$^2$ system has to download the XML files in order to perform its search and clustering functions. Therefore, the first notion is to use that systems downloading process to capture the XML abstracts when they are first encountered, eliminating steps 2 and 3 from the previous process. An important point to note is this: when those abstracts are fetched, they are all fetched at once rather than one at a time. The new process is this:

1. Create a list of query source terms(Appendix C)
2. Feed that list to a software application which performs search and clustering to collect cluster and abstract documents
3. Convert each abstract to a JSON document
4. When searching is completed, decorate each JSON document with additional features.

A Java project was created to test this idea: Carrot2PubMedStudyEngine.

## Carrot2PubMedStudyEngine

The study engine performs a combinatoric search and clustering on each query. The algorithm is this:

### Preliminary Steps

```
Read each Predicate and Concept file into a list of query terms.
```

### Search

```
For each query source term
      Run Query on that term
      For each query term
            Run Query on query source term + query term + i
            Increment i
```

The Carrot$^2$ system limits the number of documents to cluster to 150, and Carrot2PubMedStudyEngine seeks to fetch up to 20,000 documents per query, so, it is possible for there to be many cluster XML files per query. For that reason, a counter (“i”) is appended to cluster file names.

In all cases, if a cluster file exists, the query is not repeated. In all cases, if a PubMed abstract XML file exists, it is not saved again. It is possible that a given cluster document will not result in any new abstracts to download given that the referenced documents were already fetched by way of other clusters.

As descrobed, Carrot2PubMedStudyEngine is currently built to process a list of queries in a batch mode. What about the *never-ending learning system*?

## Towards a Never-Ending Learning System

Let us examine the larger processes of OpenSherlock. Then, we look at incorporation of Carrot2PubMedStudyEngine into that process to automate learning.

### OpenSherlock Processes

OpenSherlock maintains a Topic Map which has topic representations for all important terms, such as disease names, biological processes, and so forth. OpenSherlock maintains a collection of objects called WordGrams for each term in the vocabulary, and eventually every word and phrase read during harvesting. Each word or phrase has one and only one WordGram, no matter how many times it is encountered. Each WordGram keeps a list of sentence identifiers where it was encountered, and a list of topic identifiers where the word or term serves as a label (name) for that topic. Major terms, actors in relationships, are also nodes (vertices) in a graph. When two actors are connected by some relationship, for example: `Actor A causes Actor B`, the graph will represent that relationship. In that sense, each actor might serve as a focal point in a constellation of relationships.

In general, these steps are engaged in OpenSherlock:

1. Bootstrap the system's *vocabulary*
2. Collect JSON documents based on PubMed harvesting
3. Read those JSON documents

#### Bootstrap

When a fresh TopicMap is encountered, the bootstrap process imports a collection of *ontologies* and creates both topics for each ontology class, and WordGrams for the labels. It also harvests any descriptive information for each class for later *reading* by creating an `IDocument` for it.

#### Collect Documents

Described above, but the subject of the next chapter, Automating OpenSherlock.
OpenSherlock performs this process

```
For each JSON document (as described above)
      Create an instance of IDocument, a specific topic which is then
      later harvested by reading.
```

#### Read Documents

This is the primary harvesting process. It entails this process:

```
For each IDocument that has not been processed
```

```
Collect all sentences in the document (performed on a
paragraph-by-paragraph basis)
Break the sentence into individual words
For each word and phrase (up to 8 words per phrase)
      Convert to WordGram
Parse sentence with LinkGrammarParser to determine sentence
structure
Study sentence structure to find actors and relations
Update Topic Map for known actors and relations
Where actors and relations exist
      Update the graph
```

Overall, these processes still entail human intervention. What would it take to automate learning?

### Automating OpenSherlock

Consider these observations:

- The process of clustered search returns many opportunities to discover new terms, and new relations.
- The process of bootstrapping, itself, creates many opportunities for clustered search.
- New opportunities for clustered search can be addressed through automation.

All of which is to suggest that each time a new topic is created in the TopicMap, which also means a new Node is created in the graph, that new object's name can be fed back to Carrot2PubMedStudyEngine which would be enhanced to operate in a threaded, automatic mode rather than batch mode. That process then creates new clusters and PubMed XML documents which then must be harvested. Let us look closer at OpenSherlock's processes.

For biomedical terms, OpenSherlock consults DbPedia [3]. If the query results in a hit, then a new IDocument is created for harvesting. The system is constantly looking for ways to *learn more* about each new *topic* it encounters. Adding clustered search on every new term continues that process, presumably in a *never-ending* way.

# Appendix A: Predicates

absorbed by
absorb
absorbs
absorbing
accelerated by
accelerate
accelerates
accelerating
accumulated by

accumulate
accumulates
accumulating
acetylated by
acetylate
acetylates
acetylating
act as
acted as
acts as
acting as
activated by
activate
activates
activating
adapted
adapts
adapting
added by
add
adds
adding
affected by
affect
affects
affecting
aggregated by
aggregates
aggregating
altered by
alter
alters
altering
amplified by
amplify
amplifies
amplifying
arrested by
arrest
arrests
arresting
associated with
augmented by

augment
augments
augmenting
bind
binds
binding
blocked by
block
blocks
blocking
catalyzed by
catalyze
catalyzes
catalyzing
caused by
cause
causes
causing
changed by
change
changes
changing
conjugated by
conjugated with
conjugate
conjugates
conjugating
connected
connect
connects
connecting
considered as
consumed by
consume
consumes
consuming
correlate with
correlated with
correlates with
created by
create
creates
creating

deacetylated by
deacetylate
deacetylates
deacetylating
deactivated by
deactivate
deactivates
deactivating
decelerated by
decelerate
decelerates
decelerating
decreased by
decrease
decreases
decreasing
depend
depends
depending
determined by
determine
determines
determining
desensitized by
desensitizes
desensitization
desensitizing
disabled by
disables
disabling
displays
down regulated by
downregulated by
downregulates
downregulating
down regulates
down-regulated by
down-regulates
down-regulating
driven by
drives
driving
enabled by

enables
enabling
enhanced by
enhances
enhancing
extended by
extend
extends
extending
expressed by
express
expresses
expressing
formed by
form
forms
forming
found in
gated by
gate
gates
gating
governed by
govern
governs
governing
immobilized by
immobilize
immobilizes
immobilizing
impacted by
impact of
impact
impacts
impacting
impaired by
impair
impairs
impairing
impinged
impinge
impinges
impinging

implied by
imply
implies
implying
improved by
improves
improving
inactivated by
inactivate
inactivates
inactivating
increased by
inactivate
increases
increasing
induced by
induce
induces
inducing
inhibited by
inhibit
inhibits
inhibiting
intensified by
intensifies
intensify
intensifying
involved in
involved with
involvement in
involvement with
involve
involves
involving
linked with
link
links
linking
lowered by
lower
lowers
lowering
mediated by

mediate
mediates
mediating
mobilized by
mobilize
mobilizes
mobilizing
modified by
modifies
modify
modifying
modulated by
modulate
modulates
modulating
oxidized by
oxidize
oxidizes
oxidizing
phosphorylated by
phosphorylate
phosphorylates
phosphorylating
possessed by
possess
possesses
possessing
precluded by
preclude
precludes
precluding
prevented by
prevent
prevents
preventing
produced by
produce
produces
producing
promoted by
promote
promotes
promoting

protected by
protect
protects
protecting
raised by
raise
raises
raising
recognized as
recognized with
reduced by
reduce
reduces
reducing
regarded as
regulated by
regulate
regulates
regulating
released by
release
releases
releasing
removed by
remove
removes
removing
replenished by
replenish
replenishes
replenishing
replenishment
restored by
restore
restores
restoring
respond
responds
responding
sensitization
sensitisation
sensitise
sensitize

sensitised
sensitized
sensitised by
sensitized by
sensitises
sensitizes
sensitising
sensitizing
strengthened by
strengthen
strengthens
strengthening
suggest
suggests
suppressed by
suppresses
suppressing
synthesized by
synthesize
synthesizes
synthesizing
targeted by
target
targets
targeting
thought of as
transferred by
transfer
transfers
transferring
up regulated by
upregulated by
upregulates
up regulates
up-regulates
used by
use
uses
utilize
utilizes
weakened by
weaken
weakens

weakening

## Appendix B: Concepts

cancer
dementia
alzheimers
parkinson's
tremor
inflammation
brain
heart
neurodegenerative
alloimmune
autoimmune
myeloproliferative
oxidative stress
chocolate
coffee

## Appendix C: Batch Query Source Terms

#File is saved as utf8
Anthocyanins
Anthocyanin
2-hydroxyglutarate
β3-adrenergic receptors
5-htp
6-hydroxydopamine
8q24
10q21
16:4(n-3)
AAA
AA  genotype
AAT
AAT diet
ABO
acai
ACE2
acne rosacea
chocolate
coffee

acrylamide
activated T cell
activated T-cell
acute lymphoblastic leukemia
acute pain
adaptor protein
adenosine
adenosine diphosphate
adenosine triphosphate
adenosyl-cobalamin
adenylyl cyclase
adhesion molecule
adipokine
adiponectin
adipose tissue
adp glucose pyrophosphorylase
AG  genotype
AG  ratio
Akt kinase
alkaline diet
allergic rhinitis
allicin
aloe emodin
aloe vera
Alopecia Areata
alpha-1 antitrypsin
Alpha-synuclein
Alzheimer
Alzheimers
Alzheimer's
AMP
AMPK

# References

[1] Nell: Never Ending Language Learning
http://rtw.ml.cmu.edu/rtw/
[2] Carrot$^2$
http://project.carrot2.org/
[3] DbPedia
http://dbpedia.org/