LiteNet User's Manual

NOTE about this document: there have been user interface changes after creating many of the images used here. Slowly, the old images are being replaced.

Background

> “Upon this gifted age, in its dark hour,
> Rains from the sky a meteoric shower
> Of facts . . . they lie unquestioned, uncombined
> Wisdom enough to leech us of our ill
> Is daily spun, but there exists no loom
> To weave it into fabric.”
>
> — Edna St. Vincent Millay

The world is interconnected, your thoughts should be too. For knowledge workers, too often their digital desktop is a dense forest of past projects, idle thoughts, bookmarks, and that rare species of information that is needed now. This density makes it exceedingly difficult for even the most experienced explorer to find what they need. But when order is applied, in the form of folder hierarchies, something vital is lost. The ecosystem of knowledge in exchange for the sterile park of navigability.

LiteNet allows you to keep your ecosystem of knowledge vibrant, but allows you to navigate it. With LiteNet, you build a map showing the connections between what you know so that you discover what you need. LiteNet; discover your next great idea.

The LiteNet platform is an exploration of the simplest possible user experience for *network thinking in the context of note taking* - a kind of personal journal for taking notes and wiring up a network of topics of interest - a personal *knowledge graph*. That network of topics is very much like an *information fabric*. Using LiteNet, you are weaving your own information fabric.

LiteNet, in its present configuration, is a *single-user* platform. It is not configured to allow signup of multiple users.

Features

What you are creating

Fundamentally, there are two objects supported by LiteNet:

- A Journal
- A Topic Map with graphs of relations among topics

How you create them

LiteNet includes the ability to record Journal Entries by three different paths, each of which has its own subheading here:

- Text
- Triple (Statement)
- URL

Active Reading

The Open University says this about *active reading*:

> *Active reading simply means reading something with a determination to understand and evaluate it for its relevance to your needs.*

LiteNet supports the ability to

- Easily capture your thoughts, be they ideas, or passages copied from some text resource
- Spend some time reading your captured thought and identifying the concepts expressed in those thoughts
- Spend more time much later wiring those concepts into a knowledge graph - a *topic map*.

It is those acts performed after capturing thoughts which promote *active reading*.

Active reading works best when you *own* the situation which motivates the thought capture process.

Active reading is a process which harvests *stocks* from the *flows* of your thinking.

Stocks and Flows

Consider this: you have flows of ideas, streams of thoughts, whatever you want to call them. You want a way to capture them quickly; you sometimes write notes on pieces of paper, into a notebook or journal, perhaps in several different places.

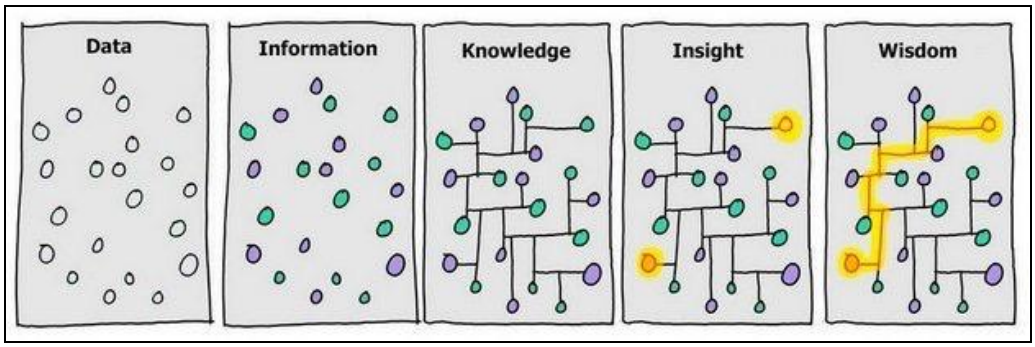
LiteNet is built to allow you to capture those thought streams in a digital environment.

Stocks are the topics of those flows, and those topics turn up in many different events - notes.

LiteNet provides the ability, with little effort on your part, to *annotate* your notes so that LiteNet will automatically respond to your annotations by creating or updating topics you identified in your notes.

LiteNet offers one more feature: the ability to create relationships among those topics, and let you wire up a graph of those topics.

Connections



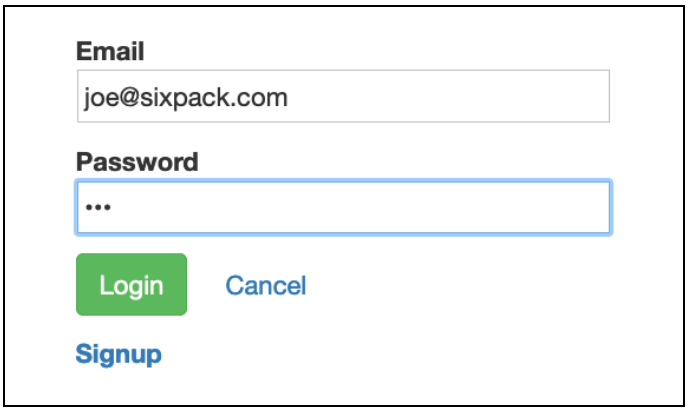
Source: Richard Feynman: https://twitter.com/proffeynman/status/1236339391909355520

User Interface

When first booted on an empty database, it looks like this:

Booted on an empty database

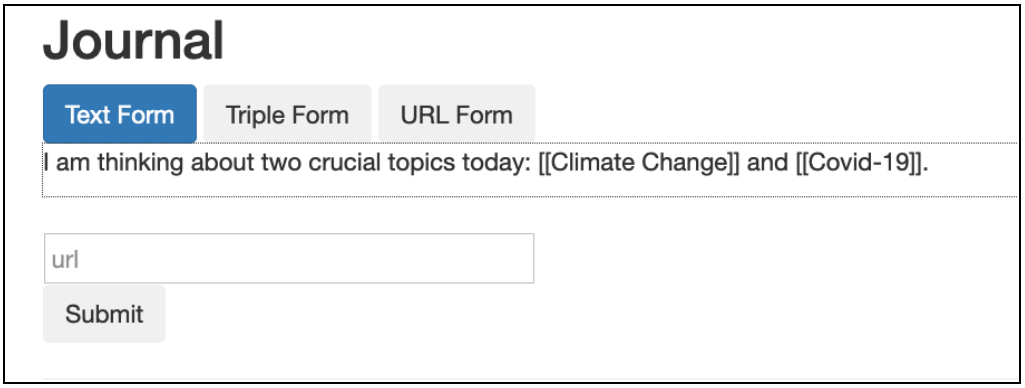
Log into the portal with credentials given in `owner.json`:



And now, you have access to controls:

A First Journal Entry

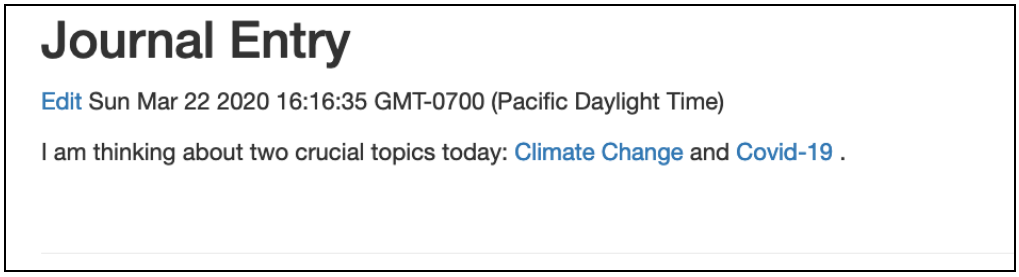
Let's create a text-based journal entry about two topics: Climate Change and Covid-19.



My Journal Entry before clicking Submit

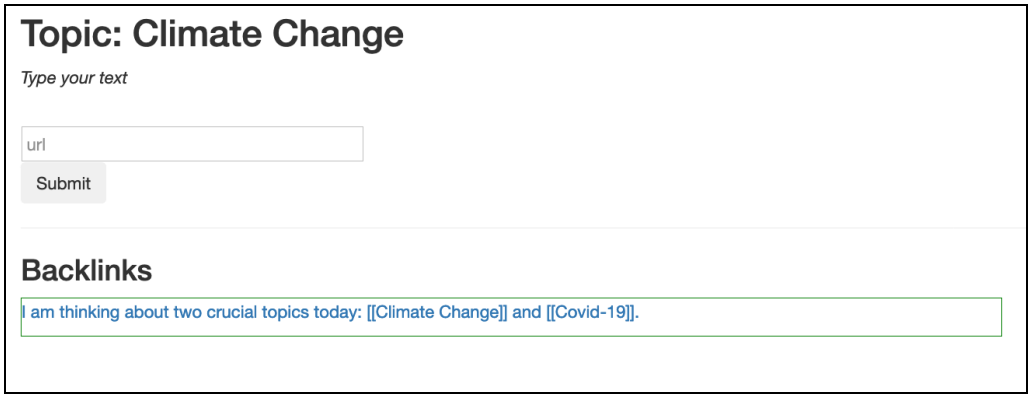
Notice that I typed in a sentence, then I made marks around the two topics: those are called *WikiLinks*. Surrounding topics with wikilinks instructs the platform to create or otherwise update those topics for me when I submit the journal entry. Let's follow what happened after submitting.

After submitting, we are taken to a view of that entry after processing:



The Journal Entry itself.

Notice that my two topics are now *hyperlinks,* ready to click. Let's click on one of them.

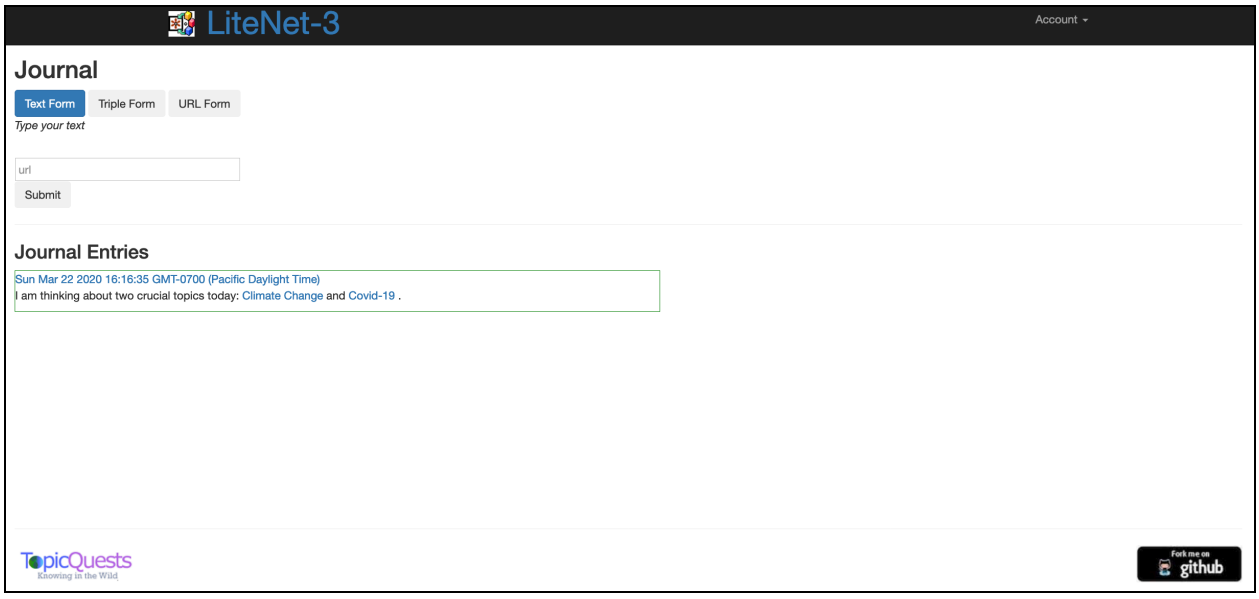


The Climate Change Topic

We observe these key factors about the topic:

- It was generated automatically because of our WikiLink in the Journal Entry
    - If the topic already exists, it will be updated with the new backlink
- It includes a *backlink* to the Journal Entry, including the text of that entry
- It has an editor where blocks of text can be added to the topic.

Finally, we observe that the landing page reflects the new Journal Entry.



Known Bugs

Topics View in Safari Browser

For reasons yet to sort out, viewing a topic in Safari will not look like the pictures here; the top Topic: <name> does not show. (20200407)

Installation

Prerequisites

Node.js

LiteNet is a *node.js* program. There are two components in play:

- The core node.js platform
- The NPM package manager

NPM is installed when you install Node.js

Node Version

Node version is important. LiteNet is being developed on node.js version 13+.
It may run properly on versions down to 11, but has been shown to fail below that.

If you already have node installed, you might have to upgrade to the latest.

Find an appropriate installer here:
https://nodejs.org/en/download/

Download from Github

The place to download LiteNet is the repository here:
https://github.com/KnowledgeGarden/lite-net-3

Recommended is simply to use the Clone or Download button and click the Download ZIP link.
Save to a folder we shall call `<litenethome>`.

Installing

From here, we presume that you are in a console (terminal).

Unzip the downloaded lite-net-3 file and then *cd* into that directory (`<litenethome>`).
There are two steps between doing that and running the system:

- Installing the necessary node.js packages with LiteNet
- Configuring the system

Installation of node packages is handled by this command line:

```
npm install
```

After installation, the directory structure of `<litenethome>` is this:

- apps/
  - lots of stuff in there
- bin/
  - www
- config/
  - vocab/
    - biomed/
      - defns.json
      - labels.json
    - climate/
      - defns.json
      - labels.json
  - config.json
  - owner.json
- data/
  - keep
  - journal
  - topics
  - users
- node_modules
- public
- routes
- views
- app.js
- LICENSE
- package_lock.json
- package.json
- Readme.md

That can take a while. Next: configure the system.

Configuring

Reference directory: `<litenethome>/config`

There are three objects found in the config directory

- `vocab/` - a directory of named predicate vocabularies
- `config.json`
- `owner.json`

vocab

Nothing to do in this directory during configuration.

config.json

This is what you see when you open that file

```
{
 "banner": "LiteNet-3",
 "vocabulary": "biomed",
 "canSignup": false,
 "isPrivatePortal": false,
 "serverPort": "13350"
}
```

- `banner:` The name which will display in a browser
- `vocabulary:` The name of the vocabulary to operate with. See Vocabularies for more.
- `canSignup`: is not meaningful at this time
- isPrivatePortal: when `false`, the portal can be surfed by anyone; when `true`, the portal demands authentication before making pages visible
- `serverPort`: is not meaningful at this time

owner.json

This is what you see when you open that file:

```
{
 "email": "joe@sixpack.com",
 "fullName": "Joe Sixpack",
 "handle": "Joe",
 "password": "joe"
}
```

- `email`: the email you choose to use at authentication
- `fullName`: the full name of the owner - plays no role at this time
- `handle`: a simple handle

- `password`: your chosen password

Note: these credentials are only in play if the user database is empty. After the user database is populated, you are free to delete owner.json, but best practices simply mean erasing the password.

Reference directory: `<litenethome>/bin`

Setting the port for this portal.

The portal comes pre-configured to port `4000`. That is found in this line in the `www` file:

```
const port = normalizePort(process.env.PORT || '4000');
```

Simply edit the `'4000'` value to suit.

Starting

In the console, type

```
npm start
```

You can now visit the portal at

http://localhost:<port> e.g. http://localhost:4000 (default)

Getting Started

Our First Entry sketch above shows the entire process of getting started, from authenticating (logging in) to creating a first Journal Entry which specifies two topics of interest.

Best Practices

It is far too soon in the life cycle of LiteNet to pontificate on best practices, but here are a few of them we have found important to keep in mind:

- Small Journal Entries, typically capturing some thought, are best.
- To the extent possible, *one topic* per Journal Entry, over time, will add value.
- It is sometimes useful to copy and paste a large chunk of text into a Journal Entry, and then go in and add WikiLinks to the topics of interest.
    - This means that *many topics* can be identified within a single Journal Entry.

- When many topics are in the Topic Map, it becomes possible to start wiring up relations by using the Triple (Statement) editor.
- From (Liu, 2019)
    - One thing at a time. No multitasking
    - When writing, attend to idea flow. Meaning, not wording. …

Fast and Slow Thinking

Some people are at their peak when thinking quickly. LiteNet helps capture those thoughts. Problem is, our quick notes are sometimes filled with typos and so forth. A useful way to deal with that is to write the notes, and *not* add WikiLinks until you save the note and go back and review it, then add WikiLinks.

In its present codebase (20200405), correcting typos will not change things, but you can still WikiLink as appropriate.

Pinning Notes

Suppose you have a note you want to *pin* (to easily find). Create a topic, let's call it `Pin`, and use that, say, at the top of any note you want to pin, e.g. `[[Pin]]`. Visit the Pin topic to see all your pinned notes.

Triples and Typeahead

Statements are triples.They are composed in three separate text boxes, each of which includes a *typeahead* feature which helps you to find an existing topic so you can avoid typos which create unneeded topics.

The typeahead feature will highlight the closest match to whatever you are typing in that text box. When it lands on one you like, click `Enter (return)` and that will be selected.

If, however, you do not see the topic you want, even though typeahead is displaying something nearby, just do this:

- Type the precise topic you want and ignore typeahead
- Do not click `Enter (return)`
- Move on to another place or click `Submit` if you are done.

Details

This chapter explains the terms and concepts, with examples

BackLink

A backlink is simply a reference to the page or object (Journal Entry) which linked *outbound* to another page (Topic). That is, backlinks refer to outbound links from Journal Entries to Topics being returned. They point back.

The process by which a backlink is created is that it is a *transclusion* of the Journal Entry into the Topic, and that object becomes a hyperlink back from the Topic to the Journal Entry.

To *transclude* a Journal Entry means that the Journal Entry never is cloned into the Topic; it is simply there by reference, and must be loaded into the Topic when a view is requested. The advantage of transclusion rather than cloning is that any change to the Journal Entry will always be reflected in the Topic's view.

Journal

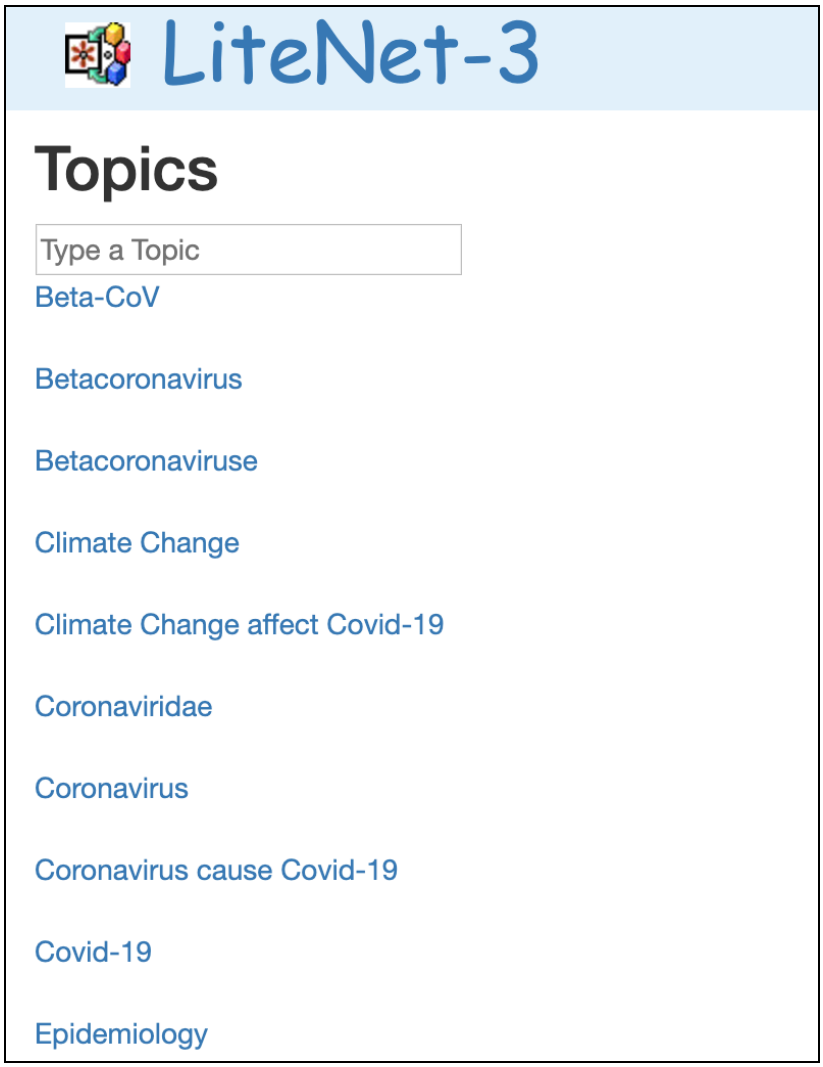
A Journal is simply a timestamped record of all notes entered into LiteNet.

Text Journal Entry

A Text Journal Entry is a note typed into the box. If a term is highlighted, a tiny Rich Text Editor bar will pop up to allow for such features as bold, italic, and so forth.

Topic Map - A Personal KnowledgeGraph

A Topic Map is a collection of Topics, which includes Relations made from Statements. Here is a list of topics based on exercises we are conducting for this document.



Partial view of the Topic Index

Triple (Statement) Journal Entry

A *Statement* is a kind of *claim* which is typically harvested from some text. That statement takes the form of a *triple*:

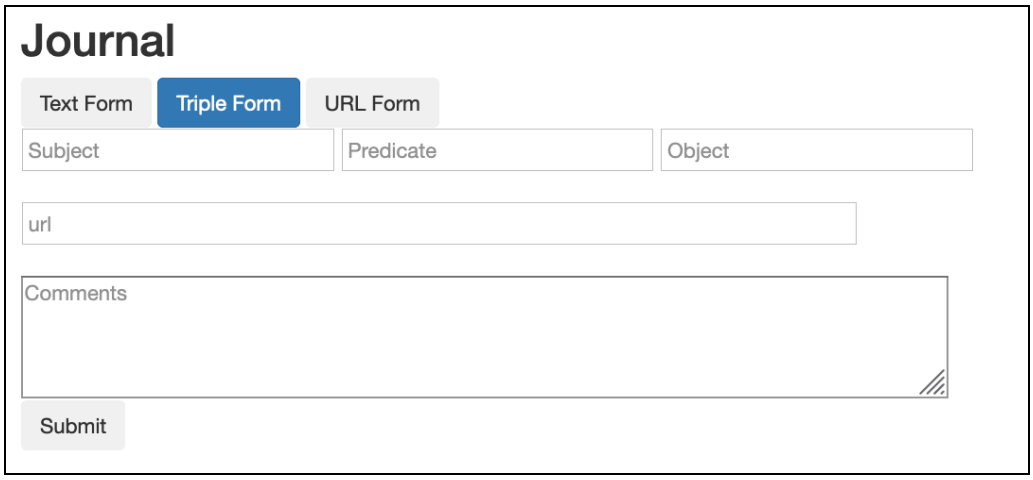
Subject, Predicate, Object

Here is an example of a statement:

Climate Change, affects, Covid-19

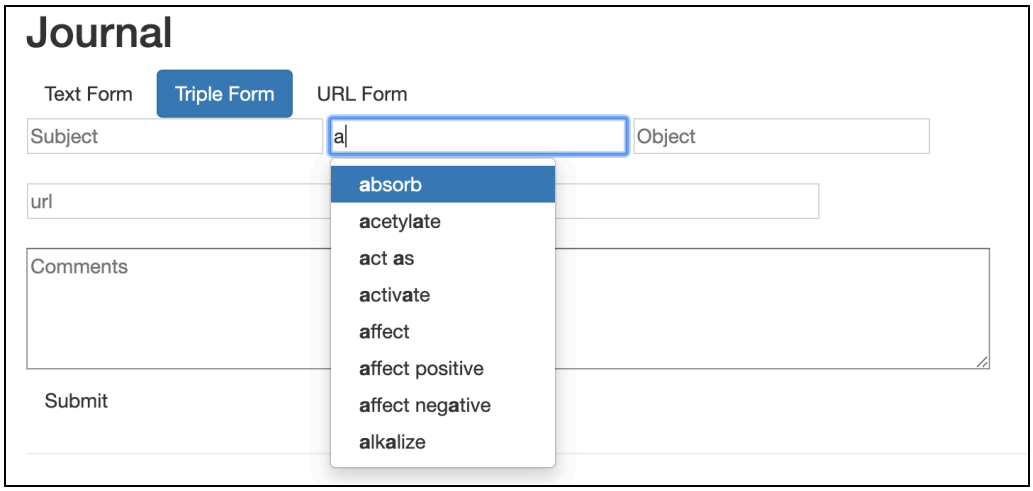
A First Statement Example

Let's create that statement by selecting the `Triple Form` on the landing page:



Empty Triple Form

Let's fill in the Predicate slot first:

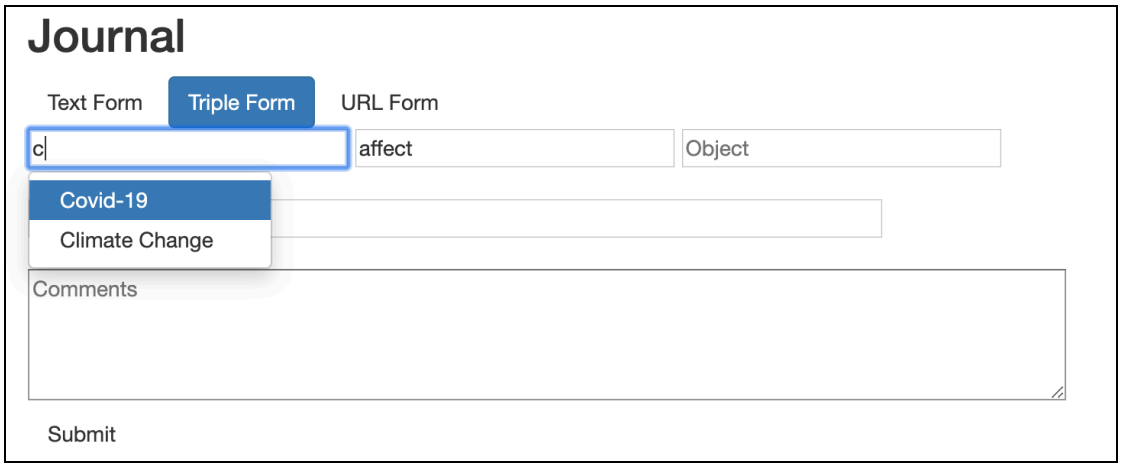


Selecting a Predicate with Typeahead

We want to select the predicate *affects* so we start by typing "a" and notice that typeahead has given us a list of hits; we will scroll down and select `affect`.

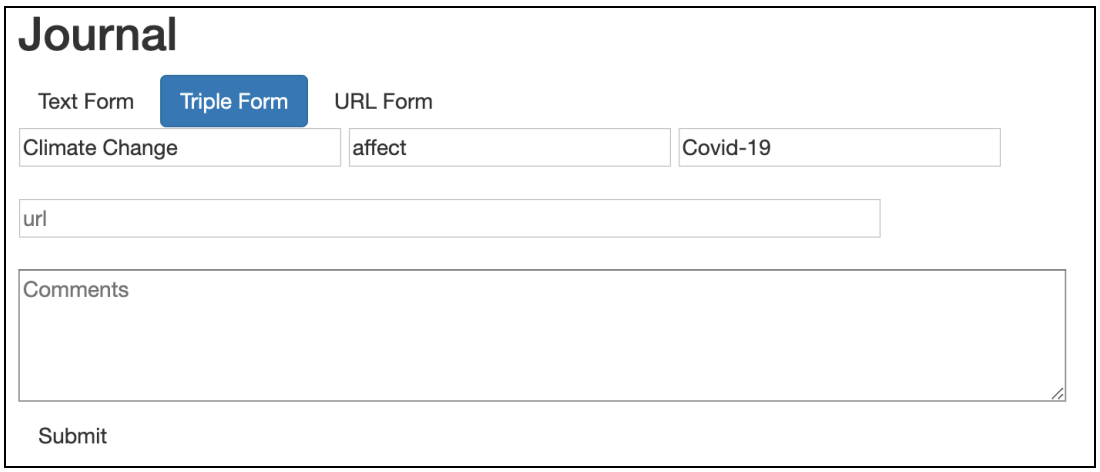
Rule: if a desired predicate is not found, it is not advised to type a new one.

Subject and Object each have a typeahead.
Note: if a subject is not found, it is permissible to create a new one. A rule of thumb: if the subject is a named entity usually spelled with Capital Letters, it's best to type it that way for the first time; after that, use of caps does not affect finding the topic.



Selecting a Subject with typeahead

Here is the completed statement before submitting it:



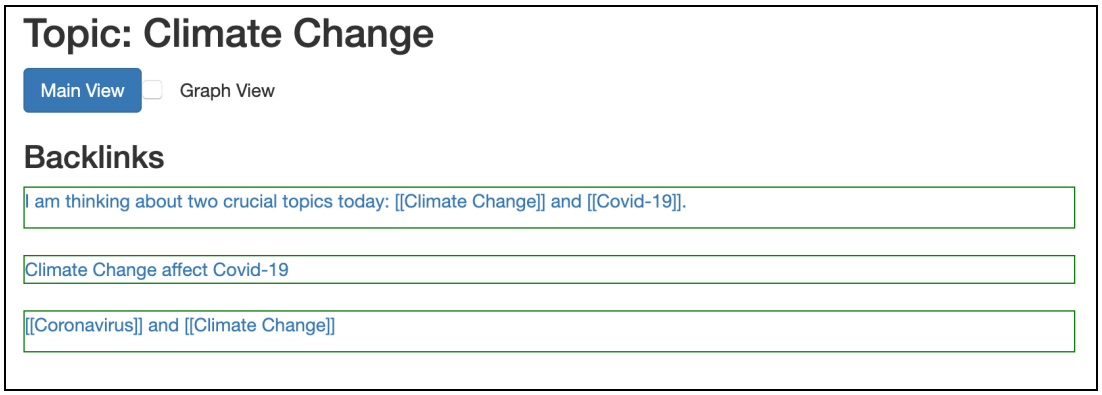
A Triple (Statement) ready to Submit

Here is where we go after clicking `Submit`:

What we know about that Journal Entry:

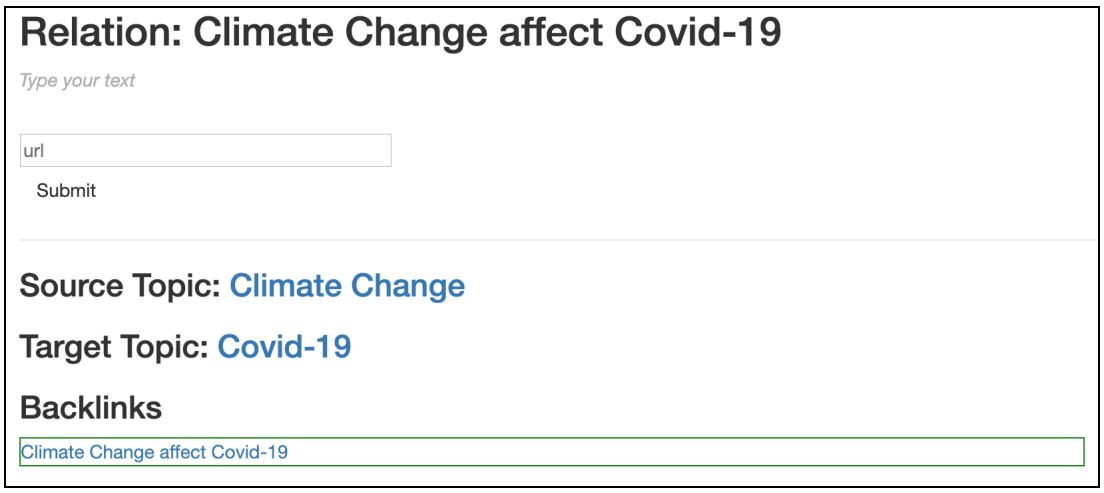
- Because we are authenticated, we can see the Edit link. If that is clicked, we have the ability to edit the text comments. Since no text comment was added to that entry, we can add one now. That text entry will be subject to WikiLinks as are all other text objects.
- Each element of the triple is *individually hyperlinked* to its topic. That includes the Predicate.

Let's look at the topic Climate Change again.



Climate Change Topic after a new backlink

Let's look at the Predicate in this statement:



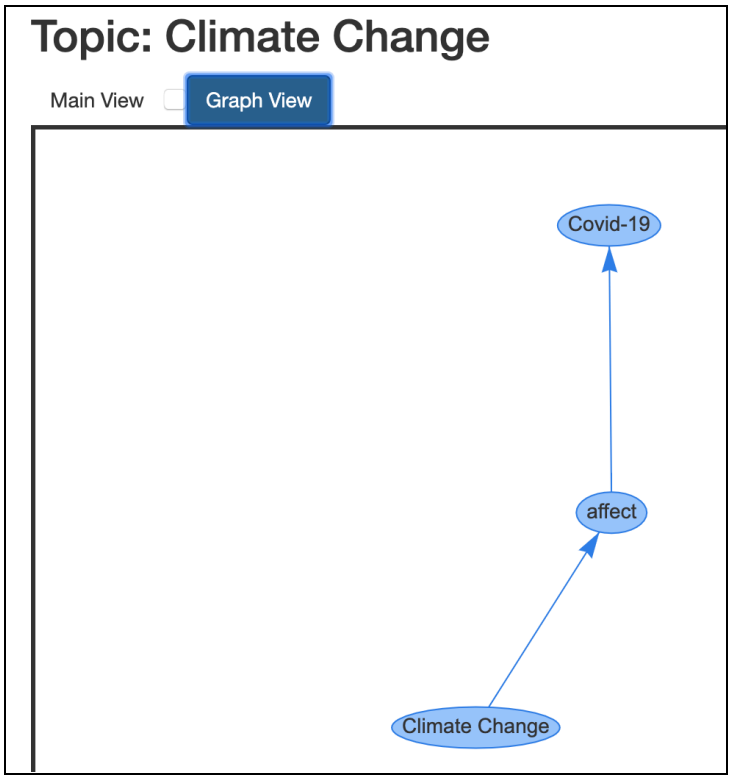
A Predicate

We observe these points about this predicate:

- It has a lone backlink to its creating Journal Entry
- It has a link to the Subject of the statement
- It has a link to the object of the statement
- As a *topic*, it carries the *full meaning* of the entire statement

The last point is important.

Let's look at the Graph of that statement:



Graph of the Climate Change Topic

A Predicate Topic carries the Full Meaning of the entire Statement

This turns out to be useful. As an example, suppose another statement wants to challenge that statement in this form:

```
Statement B, disagrees with, Statement A
```

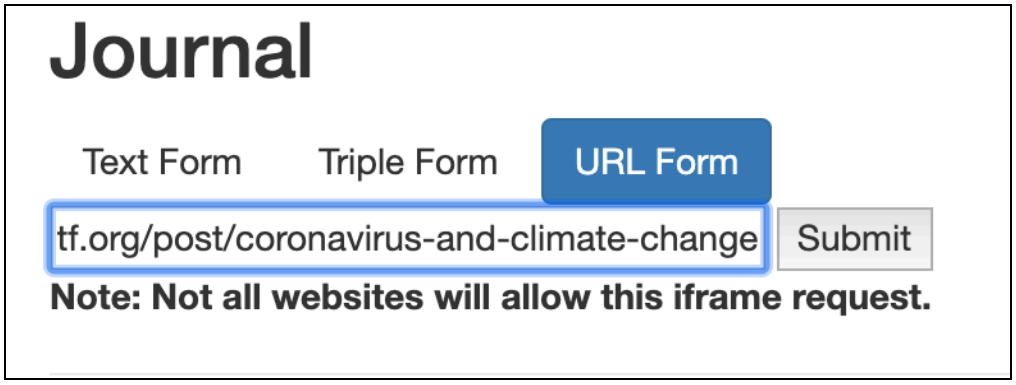
You can wire that up as a triple statement simply by using the Statement A Predicate Topic for *object*, and Statement B Predicate Topic for the *subject*.

Note: that example used a predicate of the type *coherence relation*. Coherence Relations are not yet in the vocabulary of LiteNet. It is a simple matter to add them to the vocabulary in play.

URL-based Journal Entry

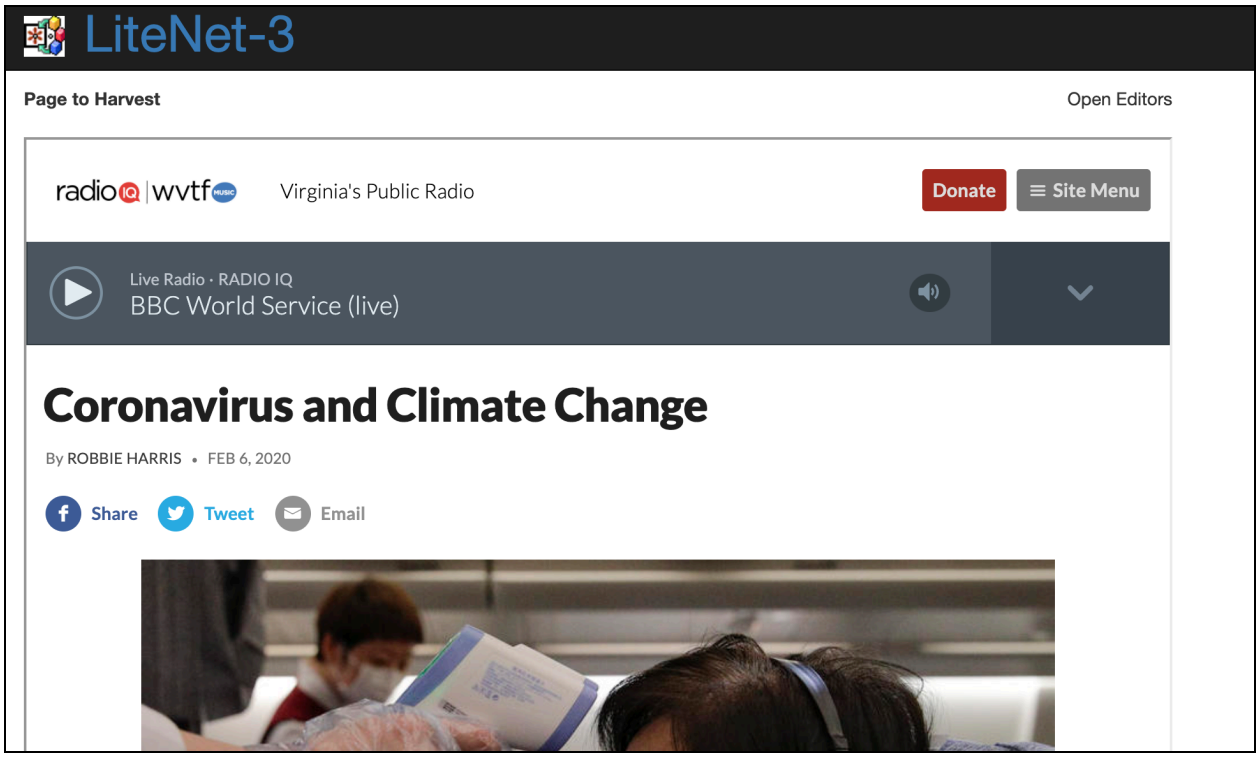
Given a URL, say, https://www.wvtf.org/post/coronavirus-and-climate-change

We paste that URL into the `URL Form`



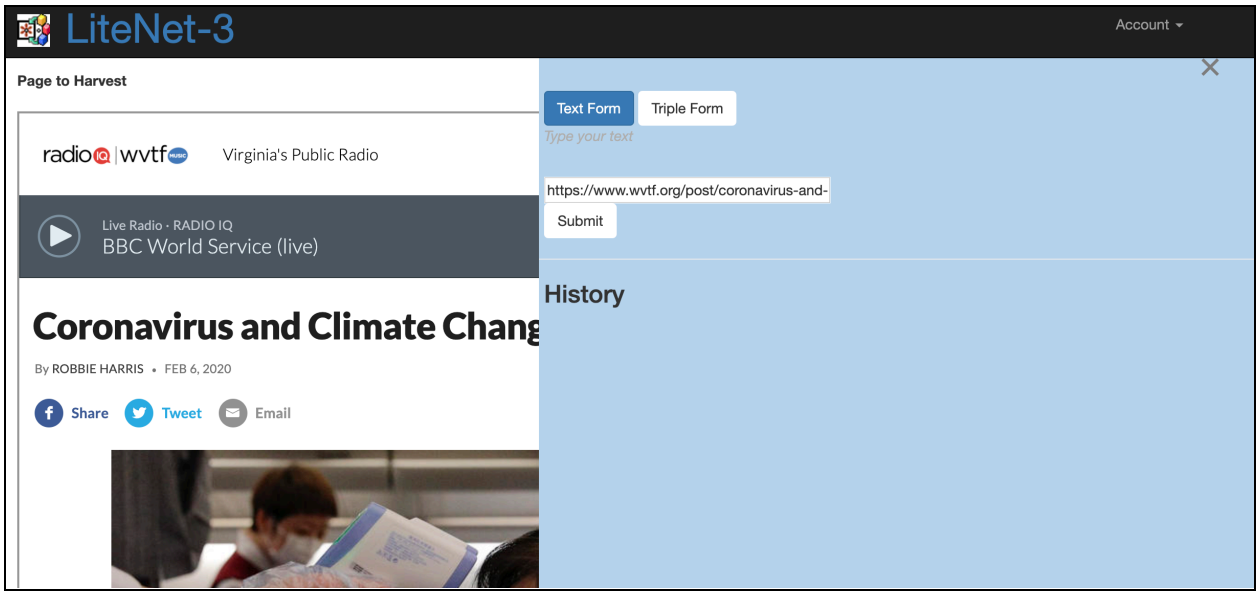
URL in the URL Form

Click Submit and you are taken to an *iframe* form which displays the document at that URL. NOTE: not all websites allow fetching by iframe.



A View of the Resource to Harvest

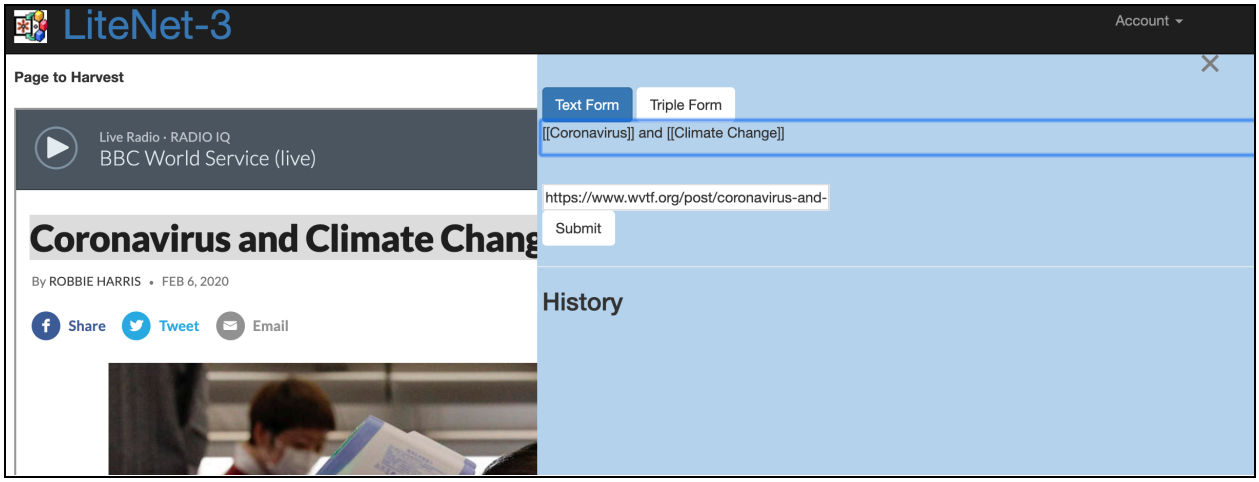
At this point, click the `Open Editors` link above the page view:



Annotation Editor with Text and Triple (Statement) Options

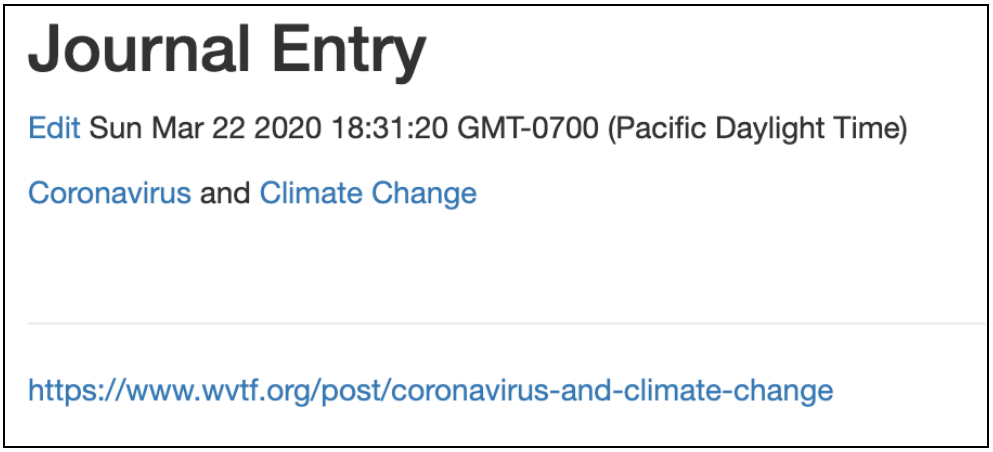
The editor sidebar can be closed by clicking the X in the upper right corner.

At this point, let's make a copy&paste text annotation. This time, it will be the document title. We close the editor, select and copy the title, then open the editor and paste it in, followed by adding WikiLinks to the two topics:

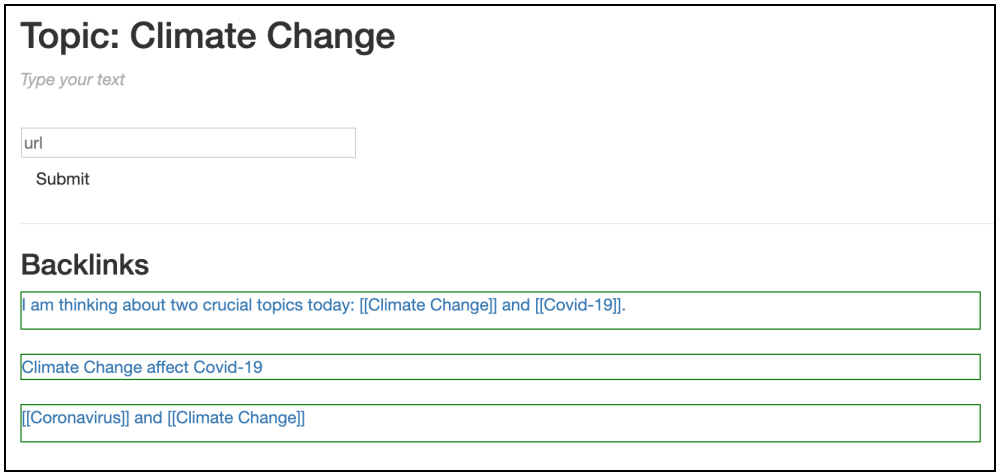


Copy&Paste title into Text Annotation and Add WikiLinks

Submit that and we now see the new Journal Entry:



If we now look at the topic Climate Change, we see another backlink:



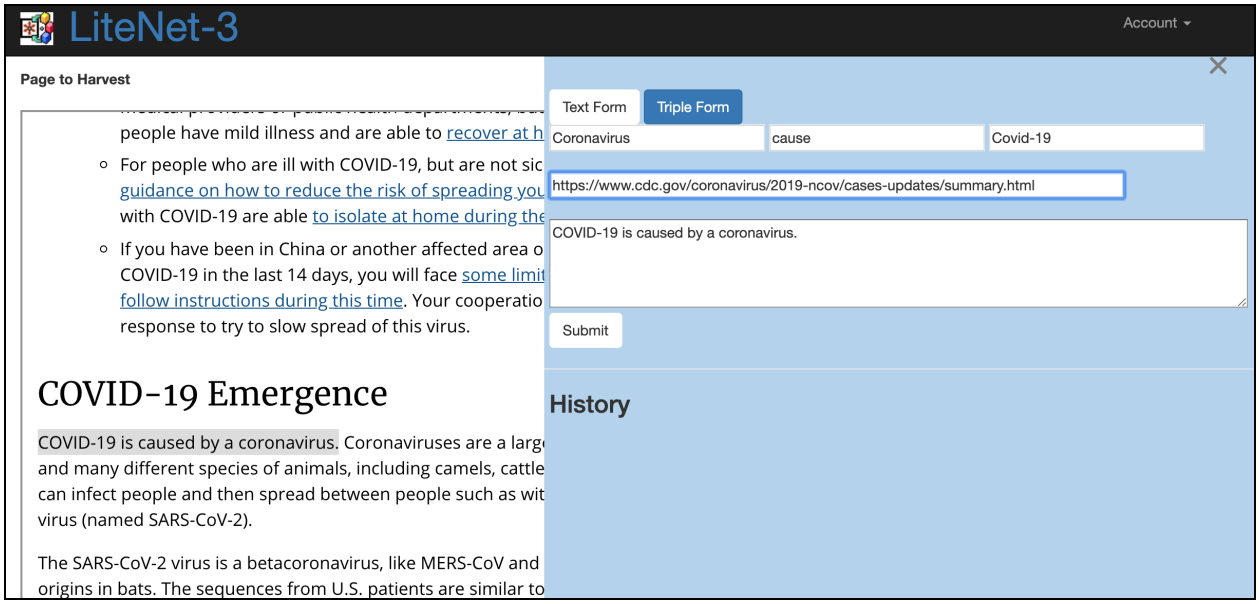
What to do When a URL is Not Available

There are URLs you might choose which block being fetched by an iframe. In that case, the process is simple:

- Remain on the landing page
- Chose one of
    - Text Form (default)
    - Triple (Statement) Form
- Past the URL there
- Return to the found page and copy what you choose to harvest
- Paste it either in the Text block (Text Form) or Comment block (Triple Form)
- Proceed from there

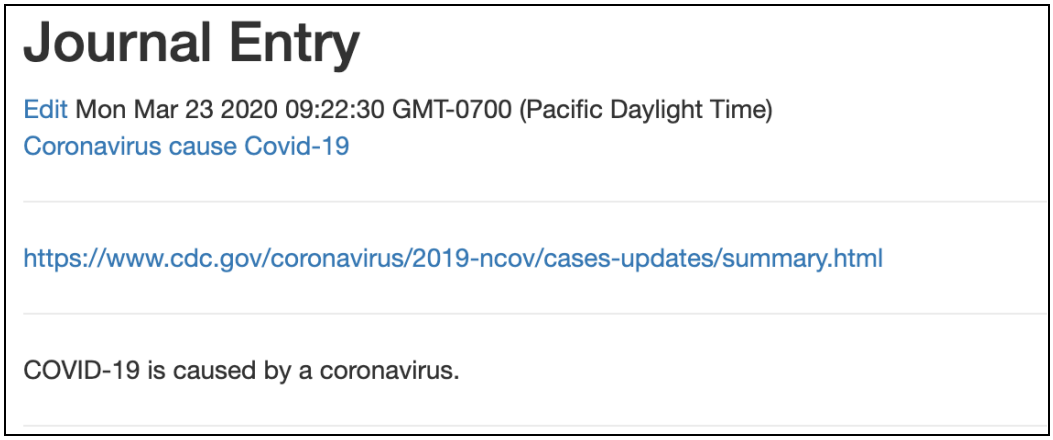
Graph Building with Statements

At this point, we note that we have two topics associated with climate change: Covid-19 and Coronavirus. It's time to do some research on the relationship between those two topics. This URL will be in play: https://www.cdc.gov/coronavirus/2019-ncov/cases-updates/summary.html as we look to find a sentence which expresses such a relationship. In the following image, we have chosen a sentence, and we harvest a statement from that sentence.

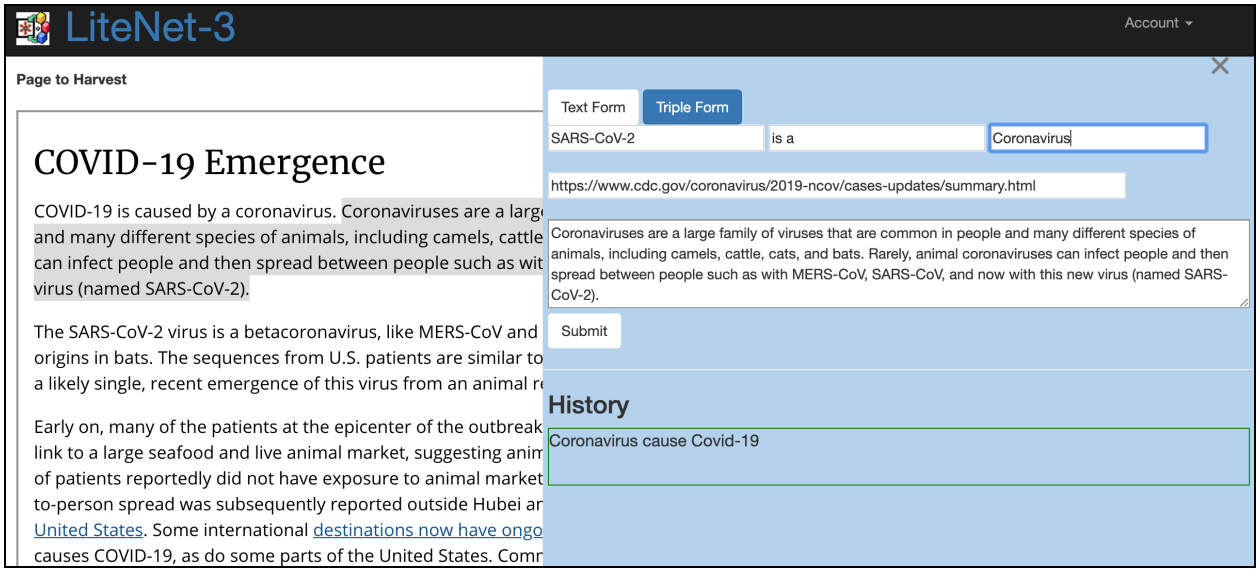


Choose a Sentence and Harvest a Statement

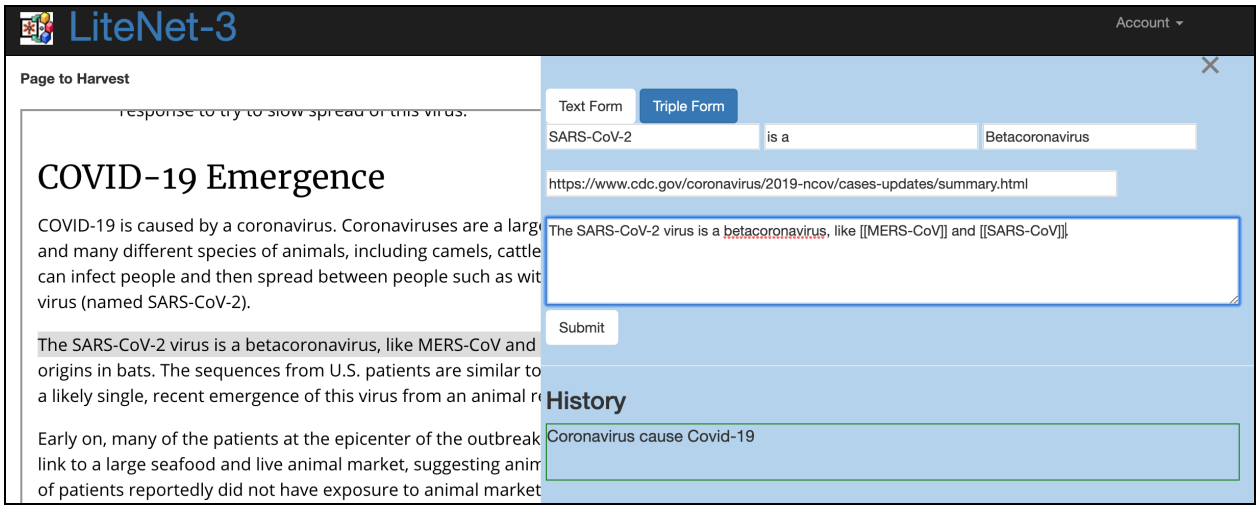
We see the new Journal Entry after submitting.



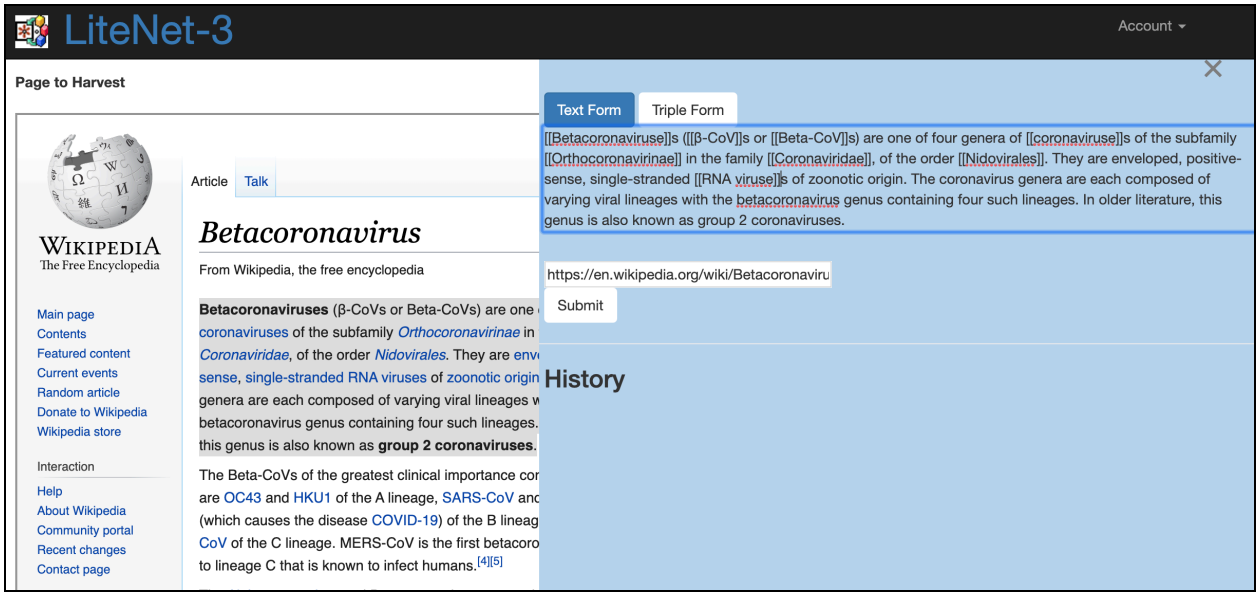
But, there is more to learn from that URL, so we return for more.



We learned that (a new topic) SARS-Cov-2 is a Coronavirus.
We are not finished harvesting from that URL.



And now, we have a new topic: Betacoronavirus. For that, we turn first to Wikipedia;
https://en.wikipedia.org/wiki/Betacoronavirus



Here, we simply copy and paste the overview as a text annotation and pluck topics out of it with Wikilinks.

Note, in that exercise, there is an attempt to stay with *singular* topics. Thus, the term Betacoronaviruses is harvested as a topic as [[Betacoronavirus]]es.[1]

Vocabularies

Vocabularies are predefined lists of predicates (verbs and verb phrases). There is one populated vocabulary, `biomed`, and one shell (unpopulated) vocabulary, `climate`. Specific vocabulary definitions are necessary in the context of using LiteNet in a collaborative setting, where Journal Entries are shared among participants.

Individuals who use LiteNet for personal use are pretty much free to do whatever they want with vocabularies, including creating *ad hoc* predicates at will.

There are two kinds of predicate vocabularies:

- Domain-specific predicates
- Coherence relations

Domain-specific Predicates

Domain-specific predicates are those which relate to specific domains. They apply to the *knowledge acquisition phase* of learning and discovery with LiteNet. Then can include many overlapping topics such as:

[1] Actually, in error, it was harvested as [[Betacoronaviruse]]s. Sigh...To correct for that, I used the Edit button and put a WikiLink on another instance of betacoronavirus.

- Causality
- Containment
- Location
- Part-whole
- etc

Predicate Design Notes

We know that sentences come in many varieties. Consider these two sentences:

1. Coronavirus causes Covid-19
2. Covid-19 is caused by Coronavirus

They both say the same thing. Sentence 1 is written in an *active voice* while Sentence 2 is not. A way to improve the readability of any graph of topics, we chose to *normalize* predicates and standardize on *active voice only*, which means the predicate `is caused by` is not available in the collection.

What that means is that the human aspect of harvesting statements requires recognizing a sentence which needs to be harvested as an active voice statement, even though it is not written that way.

Coherence Relations

Coherence relations exist for the *reflexive phase* of learning and knowledge discovery with LiteNet. While - at present- they are not added to the vocabularies, when they are, they will allow to add new information to the Topic Map. They take forms which allow to think in *narratives* like:

> This statement disagrees with that statement.
> This statement is an example of that statement.

For more on coherence relations, see:

- https://www.sfu.ca/~mtaboada/docs/publications/Taboada_Implicit_Explicit.pdf
- https://journals.openedition.org/discours/10032
- http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/repository/pdf/bateman-rondhuis97.pdf
- http://projects.kmi.open.ac.uk/scholonto/docs/SBS_DCP2002.html

Extending the Predicate Vocabularies

The system is configured such that users can create their own predicate vocabulary. The process is this:

- Create a folder in `/vocab` with a short single-word name for the new vocabulary.
- Into that folder, create a file called `labels.json`
- In that file, there is just one *key*: “`terms`”

- That key is followed by an array of comma-separated strings, each of which is the predicate itself.
- Note that in each vocabulary folder is also a file called `defns.json` - which is not presently used.

Predicate Vocabulary Heuristics

For purposes of maintaining some sense of order and structure, we apply these heuristics:

- Stay as close to the root term: e.g. `causes` → `cause`
- Try to use the *active* rather than *passive voice* of a term; e.g. `is cause by` → `cause`
-

WikiLink

A WikiLink is an *internal* link which couples two pages. There are two ways to think about a WikiLink:

- Link points to a page but the page does not point back - typical of most links in e.g. Wikipedia
- Link points to a page, and the page points back. The backward-pointing link is called a Backlink

LiteNet. is built on the principle of backlinks always in place.

Constructing a WikiLink

A WikiLink is constructed simply by surrounding some word or phrase, chosen to be a Topic, with double brackets. For example, consider this sentence:

> CO2 is a causal factor in climate change.

To turn CO2 into a topic, the sentence will look like this before submitting in a Text Journal Entry:

> [[CO2]] is a causal factor in climate change.

Note: in a future update to LiteNet, the Rich Text Editor will include a “W” button to add those brackets to any highlighted word or phrase.

Knowledge Modeling

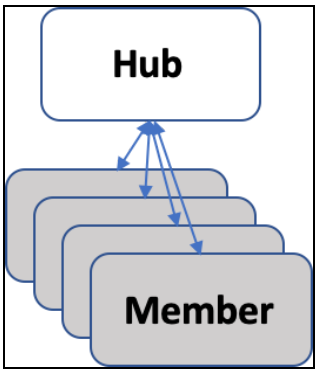
<TODO chapter ways to structure the topic map;
This is a complex topic with many options to outline>

Roadmap

Today: Single-User

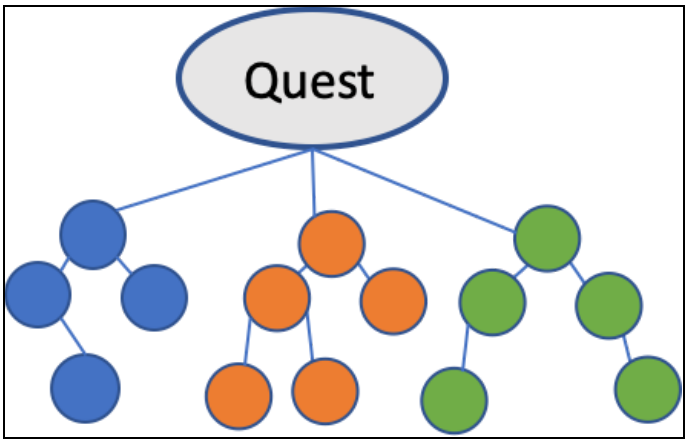
Next: Federation of Several Single-User Portals

Consider the notion of several individual LiteNet installations working together as if a *guild* in a role-playing game, or simply a team of individuals doing collaborative research by building a collectively curated Topic Map. To do that, we imagine another platform like LiteNet, but one which accepts Journal Entries from the individual installations and grows that topic map, perhaps as illustrated:



Federated LiteNet Portals

After That: Guilds, Quests, Role-playing Research Games



Further Reading

Ahrens, Sönke (2018) “How to take smart notes”. Video online at https://vimeo.com/275530205

> One of the most productive and systematic social scientists, Niklas Luhmann, worked only with pen and paper, but his note-taking system is still in many regards unmatched. It is not so much an archive, but a system for developing thoughts. In this talk I will give a brief account on the specifics of his Zettelkasten-method, typical misunderstandings, and shed some light on the reasons why it works so well from a psychological perspective.

Demski, Abram (2010). “The Zettelkasten Method” Online at https://www.lesswrong.com/posts/NfdHG6oHBJ8Qxc26s/the-zettelkasten-method-1

> My plan for this document is to first give a short summary and then an overview of Zettelkasten, so that readers know roughly what I’m talking about, and can possibly experiment with it without reading any further. I’ll then launch into a longer discussion of why it worked well for me, explaining the specific habits which I think contributed, including some descriptions of my previous approaches to keeping research notes. I expect some of this may be useful even if you don’t use Zettelkasten -- if Zettelkasten isn’t for you, maybe these ideas will nonetheless help you to think about optimizing your notes. However, I put it here primarily because I think it will boost the chances of Zettelkasten working for you. It will give you a more concrete picture of how I use Zettelkasten as a thinking tool.

Forte, Tiago, (2020). “How To Take Smart Notes: 10 Principles to Revolutionize Your Note-Taking and Writing”. Online at https://fortelabs.co/blog/how-to-take-smart-notes or https://numinous.productions/ttft/print/TTFT.pdf

> I long ago stopped reading books on note-taking.
> They were always too vague and boring, full of platitudes that had little to do with the world outside academia.
> I especially avoided “how-to” style books on the subject.
> They would often list dozens of tips and tricks that had little to do with each other. There was never an overarching system for turning notes into concrete results.
> But recently I picked up How To Take Smart Notes (affiliate link) by Sönke Ahrens.
> Ahrens is a Lecturer in Philosophy of Education at the University of Duisburg-Essen and also coaches students, academics, and professionals with a focus on time management, decision-making, and personal growth.
> It is by far the most impactful and profound book I’ve ever read on the subject. I was astounded to encounter in its pages (with uncanny similarity) many of the same principles I had discovered over 10 years of personal experience.
> This book is so full of insights that it broke my usual approach to summarizing books.

Liu, Yuxi (2019). “How to take smart notes (Ahrens, 2017)”. Online at https://www.lesswrong.com/posts/T382CLwAjsy3fmecf/how-to-take-smart-notes-ahrens-2017

> This is my rephrasing of (Ahrens, 2017, How to Take Smart Notes). I added some personal comments.

Luhmann, N. (1992). “Communicating with Slip Boxes”. In A. Kieserling (Ed.), & M. Kuehn (Trans.), *Universität als Milieu*: Kleine Schriften (pp. 53–61). Online at http://luhmann.surge.sh/communicating-with-slip-boxes

> Naturally, independence presupposes a minimal measure of intrinsic complexity. The slip box needs a number of years in order to reach critical mass. Until then, it functions as a mere container from which we can retrieve what we put in. This changes with its growth in size and complexity. On the one hand, the number of approaches and occasions for questions increases. The slip box becomes a universal instrument.

Mancini, C., Scott, D. & Buckingham Shum, S., (2006). “Visualising Discourse Coherence in Non-Linear Documents”. *Traitement Automatique des Langues*, Vol.47, 1. Online at http://kmi.open.ac.uk/publications/pdf/kmi-06-19.pdf

> To produce coherent linear documents, Natural Language Generation systems have traditionally exploited the structuring role of textual discourse markers such as relational and referential phrases. These coherence markers of the traditional notion of text, however, do not work in non-linear documents: a new set of graphical devices is needed together with formation rules to govern their usage, supported by sound theoretical frameworks. If in linear documents graphical devices such as layout and formatting complement textual devices in the expression of discourse coherence, in non-linear documents they play a more important role. In this paper, we present our theoretical and empirical work in progress, which explores new possibilities for expressing coherence in the generation of hypertext documents.

Matuschak, Andy (?). “Evergreen Notes”. Online at https://notes.andymatuschak.org/z4SDCZQeRo4xFEQ8H4qrSqd68ucpgE6LU155C

> Evergreen notes are written and organized to evolve, contribute, and accumulate over time, across projects. This is an unusual way to think about writing notes: Most people take only transient notes. That’s because these practices aren’t about writing notes; they’re about effectively developing insight: “Better note-taking” misses the point; what matters is “better thinking”. When done well, these notes can be quite valuable: Evergreen note-writing as fundamental unit of knowledge work.
>
> It’s hard to write notes that are worth developing over time. These principles help:
>
> - Evergreen notes should be atomic
> - Evergreen notes should be concept-oriented
> - Evergreen notes should be densely linked
> - Prefer associative ontologies to hierarchical taxonomies

Matuschak, Andy and Michael Nielsen (2019) “How can we develop transformative tools for thought?”. Online at https://numinous.productions/ttft

> Part of the origin myth of modern computing is the story of a golden age in the 1960s and 1970s. In this story, visionary pioneers pursued a dream in which computers enabled powerful tools for thought, that is, tools to augment human intelligence E.g., Douglas Engelbart, Augmenting Human Intellect: A Conceptual Framework (1962). One of those pioneers, Alan Kay, summed up the optimism of this dream when he wrote of the potential of the personal computer: “the very use of it would actually change the thought patterns of an entire civilization” Alan Kay, User Interface: A Personal View (1989).
>
> It’s an inspiring dream, which helped lead to modern interactive graphics, windowing interfaces, word processors, and much else. But retrospectively it’s difficult not to be disappointed, to feel that computers have not yet been nearly as transformative as far older tools for thought, such as language and writing. Today, it’s common in technology circles to pay lip service to the pioneering dreams of the past. But nostalgia aside there is little determined effort to pursue the vision of transformative new tools for thought.
>
> We believe now is a good time to work hard on this vision again. In this essay we sketch out a set of ideas we believe can be used to help develop transformative new tools for thought. In the first part of the essay we describe an experimental prototype system that we’ve built, a kind of mnemonic medium intended to augment human memory. This is a snapshot of an ongoing project, detailing both encouraging progress as well as many challenges and opportunities. In the second part of the essay, we broaden the focus. We sketch several other prototype systems. And we address the question: why is it that the technology industry has made comparatively little effort developing this vision of transformative tools for thought?