Spatial Relations in Research Papers

CS-410–Text Information Systems

Mengyu Xie, Jun Liang, Michael Wang, Adrian Bandolon

20 December 2017

Introduction

The volume of scientific literature seems to endlessly increase each year. Scientists must tackle the massive amount of existing knowledge while keeping track with the newest developments in their field. The volume and speed that scientific publication makes it almost impossible for scientists or a team scientists to read and process every scientific article. Information retrieval and computer-aided literature search can help narrow the scope of literature searches. However, these classical techniques are unable to recognize connections between literature.

Literature Based Discovery (LBD) was first proposed by Don Swanson. LBD strives to find novel connections that have not been explicitly published. Swanson's pioneering work, connecting Reynaud's disease and fish oil, was not only serendipitous but also labor intensive. He was able to make the connection only after reading hundreds of papers, and by pure chance read about Reynaud's disease and fish oil separately. Since then, several attempts at semi and full automation of the LBD process has been made. Over 30 years, the model for literature based discovery remains the same, but new techniques for extracting relationships from scientific literature has improved enough to allow more automation.

To perform LBD, Swanson introduced the "ABC" co-occurence model. In this model, A, B, and C are concepts or terms that are used to extract explicit knowledge from isolated literature. There are two main paradigms used to perform LBD, open discovery and closed discovery. In closed discovery, the user provides a start term and a target term and the system outputs a set of linking terms. In open discovery, the user inputs a start term, and system outputs a set of linking terms. Closed discovery can be used to explain correlations or observations. While open discovery is used to generate new discoveries. Figure 1 shows the two methodologies.

For this project, our team attempts to automate the open discovery model of LBD. The user will provide a term. Our app will query **PubMed**. Only the top 200 relevant abstracts will be processed. The abstracts will be tokenized and stop words will be removed using the PubMed stop word list (365 words) from the **ArrowSmith Project.** Dimension reduction will be performed using **Doc2Vec**. The vectors produced are then passed on to a plotter that uses **t-SNE** for visualization into 2D space.

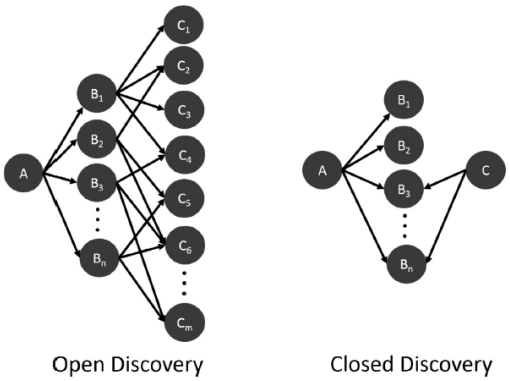


Figure 1: Open and closed literature based discovery. (From Henry and McInnes, 2017)

Methods

1. Package Dependencies

node.js	unicode	nltk
os	gensim	biopython
codecs	sys	string
xml.etree.cElementTree		

2. Components

a. **Abstract Collector** and **Processor** – queries PubMed using their **Entrez Programming Utilities**. This component is made up of three parts:

- An article fetcher, uses the *efetch* utility provided by PubMed. This returns the top 200 relevant abstracts (according to PubMed's retrieval system) in XML format.
- An article parser that extracts the article title, first author and abstract text from the retrieved articles.
- A tokenizer that produces a bag-of-words representation for each abstract. This is done by first removing punctuation and non-alphanumeric characters. Second, stop words from the **PubMed stop word list** in the ArrowSmith Project are removed, then words are stemmed using the **Snowball stemmer** from the nltk package.

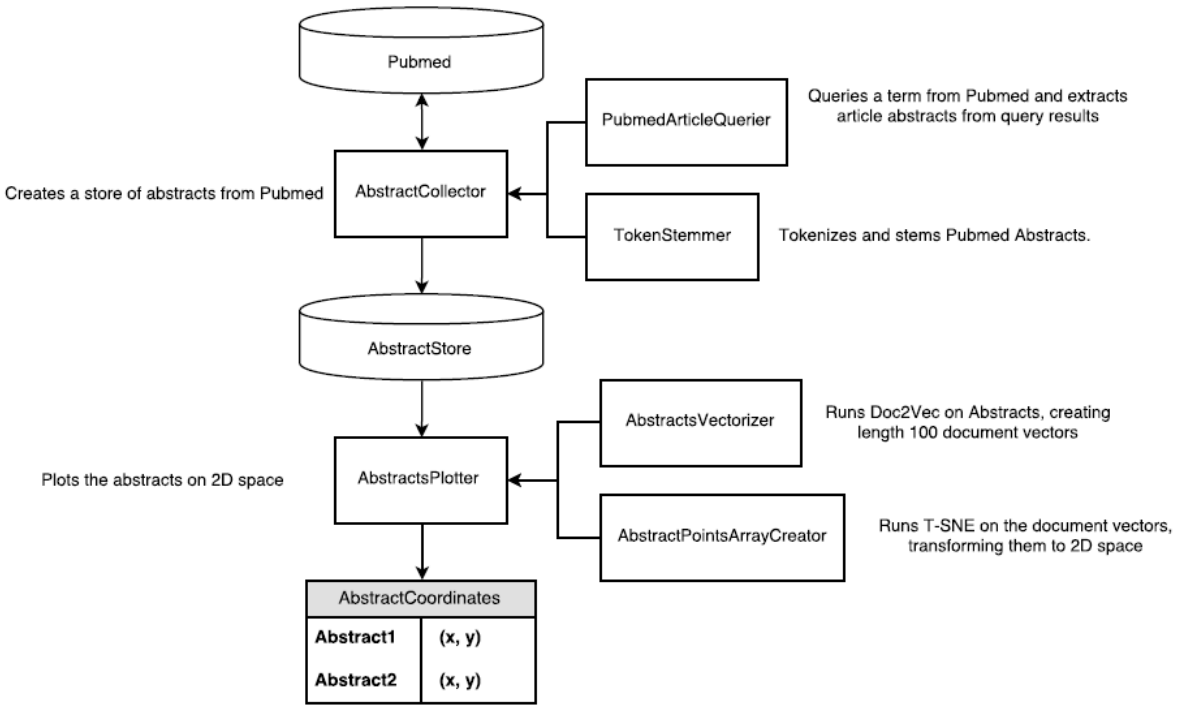
b. **Abstract Plotter** – plots the abstracts in 2D space based on the similarities of occurence frequencies of terms found in them. This is also made up to three parts:

- An abstract vectorizer that runs **Doc2Vec** on the abstracts to create document vectors. The program will return a JSON object of abstract titles, x and y-coordinates for the website to display.
- A points array creator that runs **t-Distributed Stochastic Nearest Neighbor Embedding (t-SNE)**. This transforms the document vectors, ready for plotting into 2D space.
  - **Doc2vec** is a sibling to Google's famous **word2vec** algorithm, which uses neural networks to put words in vector space. Some principles present in doc2vec and word2vec include **Continuous Bag of Words (CBOW)** and **skip grams**. CBOW tries to predict a word based on its surrounding words. Skip-grams tries to predict the words surrounding a word based solely on that word alone. In doc2vec, the equivalent of CBOW is the **Distributed Memory Version of Paragraph Vector (PV-DM)**. PV-DM tries to use the words in the paragraph to predict the values of one "paragraph vector". The equivalent of skip grams is the **Distributed Bag of Words Version of Paragraph Vector (PV-DBOW)**, where a paragraph vector is used to predict the words inside the paragraph.
  - **T-Distributed Stochastic Neighbor Embedding (t-SNE)**, is the algorithm used to map a n-length vector into 2d or 3d space. To do this, the algorithm finds pairs of vectors that are more similar to one another. It then tries to minimize the distance between two points in 2d or 3d space, and maps them onto place.
- This all packaged into a web app.

c. Team Members and Contributions

- Abstract Collector and Processor
  - Menyu Xie and Adrian Bandolon
- Abstract Vectorizer and Plotter
  - Jun Liang and Michael Wang
- Website
  - Jun Liang

3. Program Flow



4. Access and Instructions for use:

- **Access:** Our app can be accessed through **http://log1p.com**.
  - Code Repos:
    1. **https://gitlab.textdata.org/mengyux2/myproject** (Michael, Adrian and Mengyu built this whole python backend processor)
    2. **https://gitlab.textdata.org/spacial-relations-in-research-papers/spatial-relations-explorer** (Built by Jun Liang, this web-based dashboard is based on the ^ `python` processor. The processor has been modified to work with `node.js`)
- **Instructions for use:** Input a query term. Wait a little. Discover!
  - To run locally:
    a. Python
       1. Use Python 2.7.
       2. Upgrade `NumPy` and `SciPy` to the latest versions
       3. Install packages: `gensim`, `Biopython`, `nltk` and `sklearn`
       4. To test python environment, run python `Program.py <testing_term>` to verify.

    b. `Node.js`
       1. Install the latest version of `Node.js`
       2. Run `npm` install to install dependencies
       3. Run `npm` start to kick off the server