Abbreviation detection for Biomedical articles

Sonja Kenari
Lund University
Faculty of Engineering
Sweden
nat14sta@student.lu.se

Abstract

Due to the drastic increase in biomedical articles over the last few years, a way to sort through relevant articles has been done with the help of Natural Language Processing (NLP). In this project, five sub groups have worked on a work flow together to process biomedical data, specifically regarding the novel coronavirus, COVID-19. For this sub-project, the effective part of abbreviation detection in biomedical articles has been investigated. The AbbreviationDetector from scispaCy has been tested to see if it could work as a reliable text processor when it comes to detecting and replacing abbreviations with their definitions. The results indicated a decent hit rate for abbreviation detection when comparing to articles that included lists of abbreviations, but when comparing the correct definition with the matching abbreviation, lower accuracy was observed.

1 Introduction

With the amount of biomedical articles increasing at a rapid speed each day, especially during pandemic times when the world is facing the same unknown virus to understand and defeat, Natural Language Processing (NLP) is a powerful tool. With NLP, the amount of biomedical articles can be sorted properly depending on their contents and provide the most specific search for scientist and others regarding the subject. This project has been done as a part of the course EDAN70 - Project in Computer Science at Lund University, Faculty of engineering, with a focus on the "COVID-19 Open Reasearch Dataset challenge" on Kaggle, esecially on the task "What do we know about therapeutics and vaccines?". Five sub groups have worked on different parts to create a workflow together that can go through articles that are related to the novel coronavirus, COVID-19, and find the articles that can give information on what is actually known for vaccines and therapeutics for this disease by using NLP and pre-trained models. Within the search for relevant articles, abbreviation detection can become very useful, not only in COVID-19 related cases, but in general regarding biomedical and scientific articles. For when the amount of articles increase at an incomprehensible speed, the amount of newly introduced abbreviations also grow drastically. To keep readers up to speed on what each abbreviation means, the abbreviations in each article can be detected with NLP and replaced with their correct long form (i.e. the definition for the abbreviation). This way, all available prior research within a field can be comprehensible in an easier and faster way without depending on continously updating biomedical lexical ontologies. In this case, the way to detect abbreviations in biomedical articles have been investigated using the AbbreviationDetector from scispaCy. With the result from this, PubAnnotation-files have been generated for each article with the most necessary information. The accuracy for the AbbreviationDetector has then been evaluated by comparing the results from the scispaCy AbbreviationDetector to the articles that have a list of abbreviations includedas a section. If results with high hit rates were observed, it would prove that the scispaCy AbbreviationDetector can be used as a solid pre-processing step for the whole workflow of the project.

2 Background

2.1 (sci)spaCy

spaCy is a free, open-source library for NLP in Python (Honnibal and Montani, 2017). With the tools that spaCy provide, large volumes of text can be processed with the help of the pre-trained models. For cases where biomedical, scientific and clinical texts are processed, the Python package

scispaCy is used (Neumann et al., 2019). This since scispaCy models are trained on different sizes of biomedical data and word vectors.

2.2 AbbreviationDetector

By adding an abbreviation pipe to the spaCy pipeline, the AbbreviationDetector from scispaCy can be used. The AbbreviationDetector is a spaCy component and is implemented based on the algortihm for detecting abbreviations described in *A simple algorithm for identifying abbreviation definitions in biomedical text.* (Schwartz and Hearst, 2003). Since abbreviations often are introduced in articles by first writing out the long form followed by a parenthesis with the matching abbreviation within, the algorithm focuses on finding these abbreviations within the parentheses. When abbreviations are detected, the algortihm then enumerating the characters for the short form. By doing this, it can later check if these characters can be matched against the definition of the abbreviation that is prior to the abbreviation in the text. Even if all abbreviations are not capital first letter abbreviations, the AbbreviationDetector from scispaCy still requires that the first letter of the abbreviation matches the first letter of a word in the definition that the abbreviation follows.

3 Method

For this project, article data sets was accessed from Kaggle's COVID-19 project, CORD-19. In this case, a subset of 100 articles were retrieved with the help of a script implemented by a sub group in the project, along with a metadata .csv file with necessary information in separate columns for every article in the subset.

3.1 Generating PubAnnotation files

The first part of this sub project consisted of generating PubAnnotation files in .json format which was implemented in the `PubAnnotationGenerator.py` program. The information considered for each article in the subset to include in the PubAnnotation files was: cord_uid, pmcid, divid, text, project and denotations. The cord_uid and pmcid were extracted from the metadata .csv file for the subset, while the text was gathered from the .json files for the corresponding article. Divid was used as a counter to keep track of the number of sections for each article while project was given the project name for each generated file as *cdlai_CORD-19*. The denotations consisted of the results gathered with the AbbreviationDetector from scispaCy for each of the separate sections in the articles. Each detected abbreviation was saved as a dictionary with a subdictionary for its span, meaning the index number for the start letter and the end letter in the text for the article. By classing the detected short forms as "Abbreviation", they could easily be distinguished from the rest of the text as can be seen in figure 1.

3.2 Lists of abbreviations

To see which of the articles that had an abbreviation list included, so that it could be used as ground truth when analyzing the result from the scispaCy AbbreviationDetector later on, the program `scrape_abbr.py` was implemented. Since the lists of abbreviations were not included in the back matter of the .json files for the articles in the subset, a solution was implemented to web scrape each article by using the URLs found in the metadata file. To do this, Python library BeautifulSoup was used, which helps with extracting data from HTML by parsing the code so the body text gets more accessible (Richardson, 2007). When opening each URL in the metadata file for the subset, a search was made for any form of the word abbreviation (Abbreviation, Abbreviations, abbreviation, abbreviations) in the headers for the article. For the articles that gave a search hit on this, the following list of abbreviations under the header was extracted and put into separate .csv files, one for each article. Each .csv file with the results consisted of two columns, the first column with the short form of the abbreviations, and the second column gave the corresponding definition for the abbreviation. Apart from this, one .csv file was created with the doi, pmcid and pubmed_id for every article with an abbreviation list. The information was extracted from the metadata file for the subset to make it easier to find the articles with abbreviation lists in the subset further on.

3.3 scispaCy AbbreviationDetector

To get the abbreviations for the articles from their full text instead of their sections in the .json files, the full text for each article that had an abbreviation list detected from the web scraping was extracted one by one. The text was later put through the AbbreviationDetector from scispaCy to get all the detected abbreviations in the text. The result was

achieved by running `spacy_abbr.py` and its output was put in similar .csv files as for the web scraping result, which then gave that the first columns consisted of the abbreviations detected by the AbbreviationDetector, and the second column gave their definitions by using the built in *long_form* function for the AbbreviationDetector.

3.4 Evaluation

The evaluations was implemented in one program, `evaluation.py`, to get statistical output on how well the AbbreviationDetector from scispaCy worked and if it worked better for shorter sections in articles or for their full text.

3.4.1 Web scraping against scispaCy

The results for the two different .csv files with abbreviations for each article was compared by using the the result from the web scraping as ground truth. The short forms for the abbreviations in each article were compared with each other column wise, and the long forms together by using formula 1 for short forms and 2 for long forms

$$r_s = \frac{NA_{sp}}{NA_{ws}} \tag{1}$$

$$r_l = \frac{ND_{sp}}{ND_{ws}} \tag{2}$$

where r_s defines the ratio between the number of detected abbreviations from scispaCy, NA_{sp}, compared to the total number abbreviations given by the web scraping result, NA_{ws} while r_l gives the ratio between the number of correct definitions from scispaCy, ND_{sp}, compared to the total amount of definitions achieved from the web scraping, ND_{ws}. The amount of abbreviations and definitions in formulas 1 and 2 consider the exact same abbreviation and correct definition detected by the two different methods.

3.4.2 Sections against full text

By wanting to compare the best way to process a text with the AbbreviationDetector, the output of processing each text section in the article separately was compared to processing the full text of each article. This was done by extracting the number of denotations from each PubAnnotation file that was generated for articles with abbreviation lists. The number of total detected abbreviations for the sections was then compared to the number of abbreviations detected when processing the full text for the same articles. The statistical output for the comparison was given by formula 3,

$$r = \frac{\sum NA_s}{NA_{ft}} \tag{3}$$

where the ratio, r, is calculated with the fraction of the sum of abbreviations detected in the separate sections of an article, NA_s, and the number of detected abbreviations in the full text of the same article, NA_{ft}.

Hit rate	**short form** [%]	**long form** [%]
Highest	87.5	52.6
Lowest	25.0	0
Average	62.27	21.08

Table 1: Statistics for comparing the amount of different abbreviations detected by scispaCy AbbreviationDetector against lists of abbreviations in articles.

4 Results

For the PubAnnotation files, the detected abbreviations in a section was highlighted as seen in figure 1.

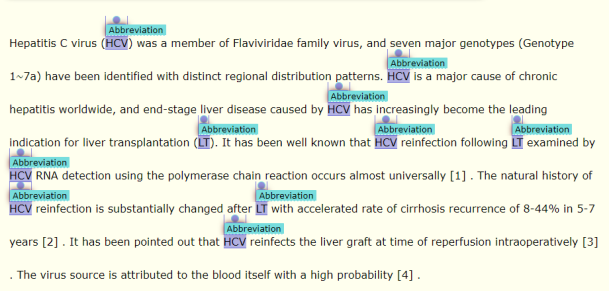

Figure 1: Abbreviations highlighted for the abstract section of the article *New insights in recurrent HCV infection after liver transplantation* (Hsu et al., 2013)

From the web scraping of the articles in the subset, 20 out of 100 was found to have lists of abbreviations included in the articles. Comparing the short and long forms for the web scraped abbreviations with the abbreviations detected in the full text with scispaCy gave the result visible in table 1.

When adding all the abbreviations detected in the sections of an article and comparing it to the amount abbreviations detected in the full texts, the average ratio with formula 3 became 32.4%. In figure 2 the statistical output can be seen in form of a histogram for the abbreviations detected in the 20 articles depending on how the text was extracted.

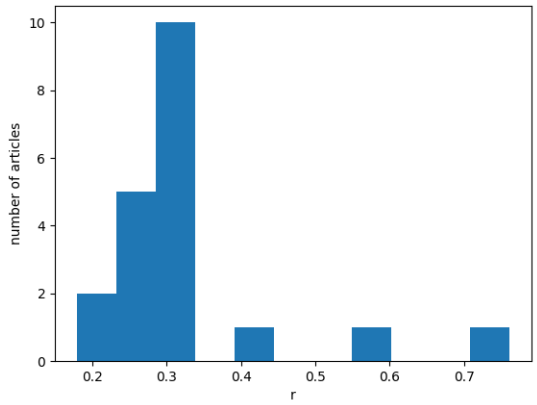

Figure 2: Ratio between the sum of abbreviations detected in each section for an article and the number of abbreviations detected in the full text directly

5 Discussion

The result in the PubAnnotation files did not give as many abbreviations in the denotations as expected, and therefore the different lengths of texts that was processed with the scispaCy AbbreviationDetector was tested. From this it could be seen that the AbbreviationDetector works better for full texts of articles instead of extracting the text from the sections within the articles and handling them separately. This since the implementation of the abbreviation detection algorithm depends on the part when the abbreviation is introduced in its short form within a parentheses. This could be a describing factor on why the amount of denotations in the several PubAnnotation files decreased when the amount of sections for one article increased, since new abbreviations and expressions rarely are presented at the end of an article. Another issue on why the average percent for the number of abbreviations detected in the different text extracting methods was so low also depends on the previous fact, since repeated abbreviations in the texts gets detected in the full text each time it occurs. This is because the AbbreviationDetector remembers the abbreviation as soon as it is introduced and then throughout the text, while it resets each time it is processed for a new piece of text, like in the case when handling the sections separately.

When comparing the results between the abbreviations detected with scispaCy and the ones in abbreviation lists, it could be seen that the average hit rate for the short form at least leaned against the higher bound. With that result it showed that the AbbreviationDetector at least finds most of the same abbreviations as given in the corresponding list. But to use this as intended when it comes to replacing abbreviations with their definitions would not be recommended when looking into the statistics for the longer form for the abbreviations. The result for the longer form leans more to the lower bound and did not give the proper definitions when using scispaCy to get the definitions for the abbreviations. This could be a downside to the implementation of the algorithm when it looks for the same first letter in the abbreviation in the words that builds the definition, since not all abbreviations, especially not in biomedical articles, use first letter abbreviations on word combinations.

5.1 Further improvements

Due to short time frame on the project, it could not be tested on a bigger data set. But running the comparison between the abbreviations detected with scispaCy on the full texts of the articles against the abbreviation lists could give more useful statistical data if the data set was bigger, relying on that 5% of the articles in bigger data set to also have abbreviation lists. With more result, it could lead up to more clarification on how to improve the long form detection for abbreviations with scispaCy and from there use its build in function to replace the abbreviations with their definitions. Further improvements would also consider on how to detect all abbreviations in different sections of a text. This to be able to include each detection of abbreviations when they are used and add them to the denotations for the sections of the PubAnnotation files.

6 Credits

I would like to thank Sonja Aits for not only coming up with this project and introducing me to the interesting field of text mining, but also for providing fruitful help and insights. And also a special thanks to my supervisor Pierre Nugues for useful inputs thoughout the project.

References

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Shih-Hsien Hsu, Ming-Lun Yeh, and Shen-Nien Wang. 2013. New insights in recurrent hcv infection after

liver transplantation. *Clinical and Developmental Immunology*, 2013.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing.

Leonard Richardson. 2007. Beautiful soup documentation. *April*.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, pages 451–462.