University of Osnabrück

Department of Cognitive Science

Dr. Phil. Peter Uhrig

Syntactic Parsing – Theory and Practice

Summerterm 2018

A Rule-Based Approach to the Entity Extraction Task on Bacteria Biotopes of the BioNLP Shared Task of 2013

06.09.2018

Ellen Schmidt

Am Salzmarkt 3

ellschmidt@uos.de

St.no.: 963682

Table of contents

1. Introduction

The amount of information available to mankind is exponentially increasing each day. Especially in the field of medicine and biomedicine the help of linguistically primed technology is essential to manage the increasing information flow and make it available to anyone. Platforms that serve as online libraries for papers and publications connected to this field are overflowing with citations and papers. MEDLINE/PubMed, which is one of the largest websites for this purpose, counted over 1,17 mio. citations for the year 2016 alone, with nearly 27 mio. citations overall. (“Yearly Citation Totals from 2017 MEDLINE/PubMed Baseline,” n.d.) To be of use to researchers or medical practitioners these information need to be organized and made more easily accessible than it would be by actually reading them. “Understanding large amounts of text with the aid of a computer is harder than simply equipping a computer with a grammar and a dictionary.” (Rodriguez-Esteban 2009)
‘Simply equipping’ now is something that has shown to be not so easy after all.
Biomedical texts cover a span from discharge summaries and consult reports over internal research reports to actual books, articles and literature abstracts and even when selecting one single medium, it is far from guaranteed that the format will be the same.

> “For example, medical centers develop their own jargons and laboratories create their idiosyncratic protein nomenclatures. This variability means, in practice, that text mining applications are tailored to specific types of text.”(Rodriguez-Esteban 2009)

To test out state-of-the-art programs and encourage in their development there are competitions, so called 'shared tasks', where the performance of parsers and NLP-programs can be tested in different situations.
For this paper the BioNLP Shared Task of 2013 was chosen, which posted challenges on information extraction in six different subject areas.(“BioNLP-ST 2013” n.d.) The entity extraction task for Bacteria Biotopes was selected and a program was written and tested on the supplied data.
The following sections will firstly look at the technical background of the application of NLP in biomedicine and the connected challenges and will then document the afore-mentioned programming project.

2. Technical Background of NLP in Biomedicine

2.1 Programs and Databases

The endeavour to use NLP systems on medical related text started with the Linguistic String Project at NYU in the 1960s. It was “one of the earliest research and development projects in computer processing of natural (i.e. human) language” (“Introduction to the Linguistic String Project” n.d.) Soon it was clear that an unspecified program like this, which just fully analyzes free text, would not achieve satisfying results in such a highly specialized field. It was especially important to tailor the programs to the unique sublanguage and grammar of the field so as to address peculiarities in both syntactic and semantic structures.

Over time a lot of task-specific solutions evolved rather than programs that tackled the whole problem at once. Subtasks include pre-processing, part-of-speech-tagging, Named-Entity-Recognition, parsing, Terminology-Mapping, analyzing the context, extract relations and events (Zhou et al. 2011, fig. 1). NegEx/ConText for example is a tool that “filters out sentences containing phrases that falsely appear to be negation phrases” (Chapman et al. 2001) from discharge summaries. MedLEE on the other hand is a program that is still in use at the Columbia Presbyterian Medical Center (Friedmann et al. 1996). Initially being created for processing radiology reports it was later extended to also handle mammography and discharge summaries (Friedman 2000), while OpenDMAP can extract information about protein transport and interaction (Hunter et al. 2008).

Before the exploring of connections between the different entities takes place, they have to be discovered and preferably mapped to a lexicon to allow comparability. For general reproducibility of health related problems the International Classification of Diseases (ICD) was developed. But to capture all the other terms related to the field, merge smaller vocabularies and allow for interoperability between systems the Unified Medical Language System (UMLS) is of much bigger importance. It is provided by the National Library of Medicine in the US and “the most comprehensive resource, unifying over 100 dictionaries, terminologies, and ontologies in its Metathesaurus” (Simpson and Demner-Fushman 2012).

2.2 Field-specific challenges

Although these systems exist, the ‘dynamic structure of scientific discovery’ (Simpson and Demner-Fushman 2012, 473) poses continuing problems in keeping track of new terms, entities and concepts. A difficulty that can be more easily resolved using e.g. UMLS is the one of synonymy, where there are multiple words for the same concept (e.g.: heart attack” and “myocardial infarction” (Simpson and Demner-Fushman 2012, 474). The opposite on the other hand, where sequences of letters have entirely different meanings, is much more challenging, most definitely context dependent and might at least be partly owed to the general inclination towards abbreviation of long and complex terms. Furthermore there might be non-letter arrays included in text, such as “laboratory values or vital signs” (Meystre et al. 2008). Clinical texts can also often be “ungrammatical and composed of short, telegraphic phrases. […] pregnant with short-hand […] misspellings” (Meystre et al. 2008) and so forth.

A great obstacle to develop and train such systems especially in directly patient-related fields can be the lack of appropriately anonymized or at least de-identified data, since the deletion of personal information in itself poses a challenge. Substantial efforts have been made as part of the 2006 ib2b de-identification challenge, where “Uzuner et al. [...] created a corpus of 889 de-identified and “re-identified" (with realistic surrogates) discharge summaries” (Meystre et al. 2008).

2.3 Shared Tasks

To find solutions to all of these problems a certain form of community challenge has established itself, especially in the field of Natural Language Processing. These so called ‘shared tasks’ are initiatives of the community supported by universities or similar related or interested sponsors. The organizers collaborate on setting a certain task, usually available to any party interested, and then continue on releasing sample, training, test and finally evaluation data as well as collect papers from the participants. Most of the time a workshop is organized at the end to portray and discuss the results. For maximum efficiency these are ideally coupled with international conferences on the same or greater topic (Biemann et al. 2013).

Looking back on the last ten years it can be said, that the competition has not only helped to develop the approaches to the tasks at hand, but also improved on handling the difficulties in the preparation process.

> “Because task data preparation and its sharing is critical for challenge evaluations, it has resulted in advancements in many related issues, such as annotation guideline standardization, alternatives to expert annotation and annotation tool development.”(Huang and Lu 2016)

3. The Programming Project

3.1 The Task of Entity Extraction

Proposed by the Bio-NLP ST of 2013 was the challenge to extract habitats of bacteria types in short descriptive texts, mapping them to MBTO-Habitat concepts and define localization relations. Their motivation came from the fact, that there was “no comprehensive database of natural environment location of bacteria, although this is a critical information for studying the interaction mechanisms of the bacteria with its environment at a molecular level”(Bossy, Nédellec, and Jourde 2012)

Chosen for this project was the first part of the first subtask, namely to find habitat entities in the text. The data used to develop the following programs were provided on the website and contained the training-data, that comprised of .txt-files and corresponding .a2-files, which contained a counter, onset, offset and an entity or entity-phrase per line, and the development-data, which were used as a test-set. Since the original evaluation-tool, that was supplied, required the annotation with ontology concepts, it could not be used and instead a new program had to be written to compare the project-generated output to the given one in the development data.

Additionally all the .txt-files were parsed with the Standford Core NLP to obtain dependency relations in the form of .xml-files.

3.2 Approach

The idea for the programs aimed towards a rule-based strategy where the given habitat entities were traced back in the parse so as to determine the governor they depended on. This dependency could then act as a rule to finding new entities in the test-data. Searching over their parse the governor would work as a trigger word. If this term in the test-text then also had a dependent of the rule-given relation, it would probably be an entity of interest.
The project was split up into two main tasks, namely creating a list of rules by using the annotated training data and then applying this rule set on the test data to extract the entities. Therefore two programs were written: rule_extract.py and entity_extract.py. Both are written in Python, were tested on Python 2.7 and require `re`, `os`, `lxml etree`, `difflib` and `math` packages as extensions. Additionally two smaller programs had to be written. Firstly, cleanup.py, to delete double rules from the rule-list and secondly the evaluation program, evaluate.py, to calculate recall, precision and F1-score on the results. All programs can be obtained from the GitHub repository https://github.com/ellschmidt/SyntacticParsingMIE.

3.2.1 rule_extract.py

This program takes as input an .a2-file and finds in every line the entity-phrase, its onset and offset. With the help of regular expressions it then looks up the entity in the .xml-file that was stripped of any but the enhanced++ dependencies to find the right index. In situations, where there is a conjunction in the phrase, these dependencies will mark the governor of the first conjunct also as the governor of the second, while with basic dependencies, the first conjunct would become governor of the second, which is not feasible as a rule. The search of the extracted entities inside the parse came with problems that are quite common in linguistics. The program had to be able to manage apostrophes, brackets, hyphens (and the parting of words at those) and special characters such as ‘%’. The tool of regular expressions were again of great help, since it was easy to look for these specifics and then either keep or delete the brackets, only use the number in front of the percentage-sign (since the sign was dependent of the number), delete apostrophes and the additional ‘s’, and also

look for the word without hyphens and appendages. Finding the indices was necessary to select the correct dependencies. Those were analyzed to work out the head of the entity-phrase and its relation to its governor, which were then written into the output-file in the format of ‘relation + governor’ with one rule per line.
A couple of rules had to be deleted in order for entity_extract.py to run on all the files, because that may have been the extracted relation, but none that can be generalized. Those were dependencies including roots and determiners, and further ‘% + case’, ‘stratified + dep’ and ‘stratified + cc’.

3.2.2 cleanup.py

This is just a small program which should be run after rule_extract.py as it looks through the whole keywords.txt-file and deletes rules that occur more than once.

The program also deletes the temporary ‘output.xml’. If both main programs are run in the same folder, then it can also be executed again after entity_extract.py, although the deletion can of course also happen easily by hand.

3.2.3 entity_extract.py

The input here is a text-file with pure text, although a parse of this text in the form of an .xml-file has to be present in the same directory. The reason that the program is not called directly with the .xml.file is firstly to ensure that the right .xml-file is searched, and further to allow a possible redirection of the .txt-file inside the program to another parser if wanted (fitting parser-output has to be ensured).
Again the xml-parse tree is stripped from all but the enhanced++-dependencies and is joined up to one string. With the regex command `re.findall()` it is now possible to find every word that matches to the current rule instead of just the first one. The flag `re.DOTALL` has to be raised to account for any remaining newlines, that although the xml has been concatenated to one string, are still existent. Looking for the rule in the text will find as a word, but often entities can be phrases. That is why it is necessary to also look for further subtrees. A recursion is the easiest way to do that. Instead of writing a whole new function,

a for-loop and appending to the list over which is looped during the loop serves the same purpose in this case. So every found dependent is added to be looked up for its own dependents. This can get stuck, if the entity is the head of a whole relative clause. The program will add up enormous lists, which will take a long time to search and have no purpose, since the relative clause in this set-up is not part of the entity-phrase. Therefore another search is launched in the results of every subtree search to call attention to relative pronouns such as “where”, “which”, “that”, “who”, and the comma that precedes them. The program then goes on to delete the comma, the pronoun and every word that comes next in the sentence for further searching in this loop.

It is also very important to take care of the fact that not every word has a subtree without stopping the whole program. The solution to this will at the same time be the anchor to this customized recursion. The way to do this in Python is the `try` and `except` keywords, where the program will execute the try-clause until an expected and specified error occurs in which case it will execute the except-clause. In this program the error raised will be `IndexError`) and the except-clause will state to simply continue with the next word in the list. When the entity-phrases are finally found they are stripped of determiners specifically parsed bracketing. Further if there are concatenations with “and” in the phrase, it will be split into two phrases to output.

Before all found entities can be printed into the output file, they have to be sorted in order of appearance in the text, since until now they have been in order of the rules from the rule-file. For this purpose the on- and offset of the phrases are determined again by search with regular expressions, and the list of found entities sorted accordingly. Eventually the whole list of results is printed in the output .a2-file appropriately formatted.

3.2.4 evaluate.py

The evaluation can be called with any file-type as long as it has the same name as the .a2-file that needs to be checked. The comparison data is expected to be in the same directory, bearing the same name plus the addition of “ (2).a2”. This can easily be achieved when copying them in the output directory containing the prediction data. The program will then go on trough every line in the originals and check for matches in the generated file, while keeping track of how many lines there are in the file. Matching lines are detected with the help of the `difflib SequenceMatcher`. Any value above 0.6 is considered a close match (“Python Docs Sequence Matcher” n.d.). With these information it then goes on to calculate recall, precision and the F1-measure and writes the results into a txt-file, each line also containing the extension-stripped name of the file and the number of matches.

3.3 Evaluation

Firstly it has to be mentioned, that the calculations in the evaluation program turned out to be inaccurate, since they did not account for double matching of prediction lines. This is also the reason why after a certain threshold the measures reach 1.0 and go even beyond.

Nevertheless a few information can be drawn from the evaluative data. It is visible that that all three measures start of with higher values on the training data (cf. Figure 1), than on the development data (cf. Figure 2), which is easily understandable, since the rules were derived from the training files.

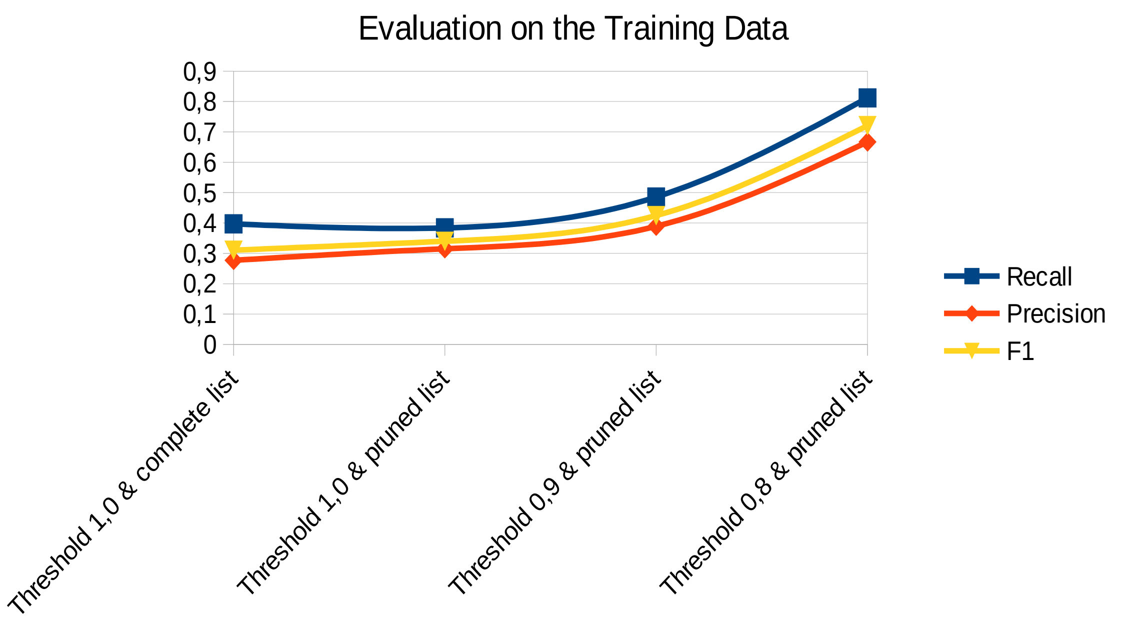


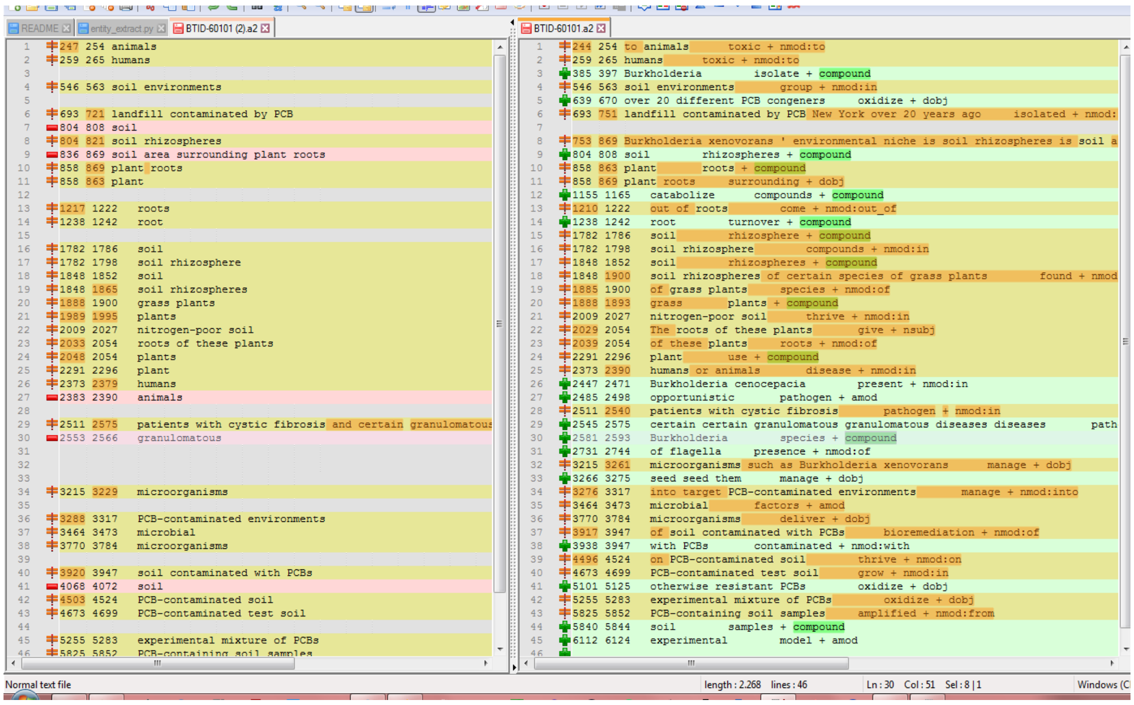
Figure 1: Evaluation of the Training Data

Secondly after comparing the training date predictions and originals a few rules were found to be inept (cf. Table 1) and the list was pruned. This lead to better results as can be seen in Figure 1.

rule	**Words that were selected as entities**
disease + amod	deadly devastating
bacteria + amod	aerobic heterotrophic PCB-catabolizing
useful + nsubj	those traits
ranging + nmod:from	insects to humans
has + nsubj	Bacillus subtilis
have + nsubj	almost all of these genes

Table 1: Examples for inept rules

Looking at the example of the ‘compound’ dependency it can be quite difficult to exclude certain rules from the beginning, since they sometimes are as often right, as they are not. In Image 1 two files are being compared, the original on the left and the prediction on the right with the appropriate rule added in each line. Green lines in the right hand file show wrong selections of entities, while orange ones just show that they were matched, just not in the same spot. Rules containing the ‘compound’-relation are highlighted in dark green and they occur about as often with wrongly selected entities as they do with rightly selected ones.



```
247 254 animals
259 265 humans

546 563 soil environments

693 721 landfill contaminated by PCB
804 808 soil
804 821 soil rhizospheres
836 869 soil area surrounding plant roots
858 869 plant roots
858 863 plant

1217 1222   roots
1238 1242   root

1782 1786   soil
1782 1798   soil rhizosphere
1848 1852   soil
1848 1865   soil rhizospheres
1888 1900   grass plants
1989 1995   plants
2009 2027   nitrogen-poor soil
2033 2054   roots of these plants
2048 2054   plants
2291 2296   plant
2373 2379   humans
2383 2390   animals

2511 2575   patients with cystic fibrosis and certain granulomatous
2553 2566   granulomatous



3215 3229   microorganisms

3288 3317   PCB-contaminated environments
3464 3473   microbial
3770 3784   microorganisms

3920 3947   soil contaminated with PCBs
4068 4072   soil
4503 4524   PCB-contaminated soil
4673 4699   PCB-contaminated test soil

5255 5283   experimental mixture of PCBs
5825 5852   PCB-containing soil samples
```

```
244 254 to animals      toxic + nmod:to
259 265 humans      toxic + nmod:to
385 397 Burkholderia        isolate + compound
546 563 soil environments       group + nmod:in
639 670 over 20 different PCB congeners     oxidize + dobj
693 751 landfill contaminated by PCB New York over 20 years ago     isolated + nmod:

753 869 Burkholderia xenovorans ' environmental niche is soil rhizospheres is soil a
804 808 soil        rhizospheres + compound
858 863 plant       roots + compound
858 869 plant roots     surrounding + dobj
1155 1165   catabolize      compounds + compound
1210 1222   out of roots        come + nmod:out_of
1238 1242   root        turnover + compound
1782 1786   soil        rhizosphere + compound
1782 1798   soil rhizosphere        compounds + nmod:in
1848 1852   soil        rhizospheres + compound
1848 1900   soil rhizospheres of certain species of grass plants        found + nmod
1885 1900   of grass plants     species + nmod:of
1888 1893   grass       plants + compound
2009 2027   nitrogen-poor soil      thrive + nmod:in
2029 2054   The roots of these plants       give + nsubj
2039 2054   of these plants     roots + nmod:of
2291 2296   plant       use + compound
2373 2390   humans or animals       disease + nmod:in
2447 2471   Burkholderia cenocepacia        present + nmod:in
2485 2498   opportunistic       pathogen + amod
2511 2540   patients with cystic fibrosis       pathogen + nmod:in
2545 2575   certain certain granulomatous granulomatous diseases diseases       path
2581 2593   Burkholderia        species + compound
2731 2744   of flagella     presence + nmod:of
3215 3261   microorganisms such as Burkholderia xenovorans      manage + dobj
3266 3275   seed seed them      manage + dobj
3276 3317   into target PCB-contaminated environments       manage + nmod:into
3464 3473   microbial       factors + amod
3770 3784   microorganisms      deliver + dobj
3917 3947   of soil contaminated with PCBs      bioremediation + nmod:of
3938 3947   with PCBs       contaminated + nmod:with
4496 4524   on PCB-contaminated soil        thrive + nmod:on
4673 4699   PCB-contaminated test soil      grow + nmod:in
5101 5125   otherwise resistant PCBs        oxidize + dobj
5255 5283   experimental mixture of PCBs        oxidize + dobj
5825 5852   PCB-containing soil samples     amplified + nmod:from
5840 5844   soil        samples + compound
6112 6124   experimental        model + amod
```

Image 1: Screenshot of two annotation-files being compared in Notepad++ with the Compare-Plugin

Furthermore the threshold for matching the lines was decreased step-by-step and it can also be seen to increase the efficiency.

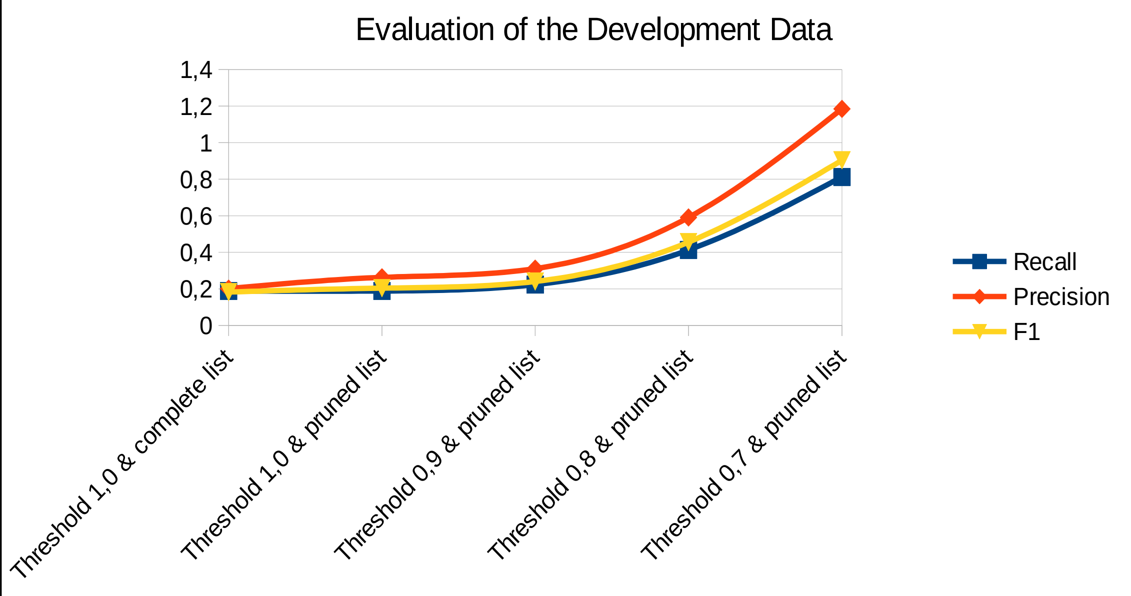


Figure 2: Evaluation of the Development Data

4. Conclusion

For the extraction of information in the biomedical field, rule-based approaches have been widely found useful. The uniqueness of the sublanguage can provide difficulties, but when taken into account appropriately can also be of great use in allowing a very fine tuning of the rules.

Overall the rule-base approach turned out to be a valid option in tackling the problem at hand, although manual pruning or further specifications inside the programs might be necessary. A complete linguistic analysis of the rules and the syntactic specifics could be very helpful. From the computational side the use of regular expressions made it comparably easy to handle known NLP problems, such as being prone to non-letter signs.
Further, the development of the programs is by far not complete. They were not revised concerning runtime minimization or computational efficiency. Also the foundations for using lemmas instead of the actual words for creating the rules were laid in rule_extract.py, but not brought to active use.
Most importantly evaluation should be repeated with corrected calculations to produce comparable results.

5. Bibliography

5.1 Papers

Biemann, Chris, Stefanie Dipper, Anette Frank, Manfred Stede, and Torsten Zesch. 2013. “Leitfaden für die Konzeption und Durchführung von shared tasks für deutschsprachige Daten.” http://www.computerlinguistik.org/LeitfadenSharedTasks161213-1.txt.

Bossy, Robert, Claire Nédellec, and Julien Jourde. 2012. “Bacteria Biotope (BB) Task at BioNLP Shared Task 2013 - Task Proposal.” Http://2013.bionlp-St.org/Tasks. October 19, 2012.

Chapman, W. W., W. Bridewell, P. Hanbury, G. F. Cooper, and B. G. Buchanan. 2001. “A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries.” *Journal of Biomedical Informatics* 34 (5): 301–10. https://doi.org/10.1006/jbin.2001.1029.

Friedman, Carol. 2000. “A Broad-Coverage Natural Language Processing System.” In *Proceedings of the AMIA Symposium*, 270. American Medical Informatics Association.

Friedmann, Carol, Lyudmila Shagina, Socrates A. Socratous, and Xiao Zeng. 1996. “A WEB-Based Version of MedLEE: A Medical Language Extraction and Encoding System.”

Huang, Chung-Chi, and Zhiyong Lu. 2016. “Community Challenges in Biomedical Text Mining over 10 Years: Success, Failure and the Future.” *Briefings in Bioinformatics* 17 (1): 132–44. https://doi.org/10.1093/bib/bbv024.

Hunter, Lawrence, Zhiyong Lu, James Firby, William A. Baumgartner, Helen L. Johnson, Philip V. Ogren, and K. Bretonnel Cohen. 2008. “OpenDMAP: An Open Source, Ontology-Driven Concept Analysis Engine, with Applications to Capturing Knowledge Regarding Protein Transport, Protein Interactions and Cell-Type-Specific Gene Expression.” *BMC Bioinformatics* 9 (January): 78. https://doi.org/10.1186/1471-2105-9-78.

Meystre, Stéphane M., Guergana K. Savova, Karin C. Kipper-Schuler, and John F. Hurdle. 2008. “Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research.” *Yearbook of Medical Informatics* 17 (01): 128–144.

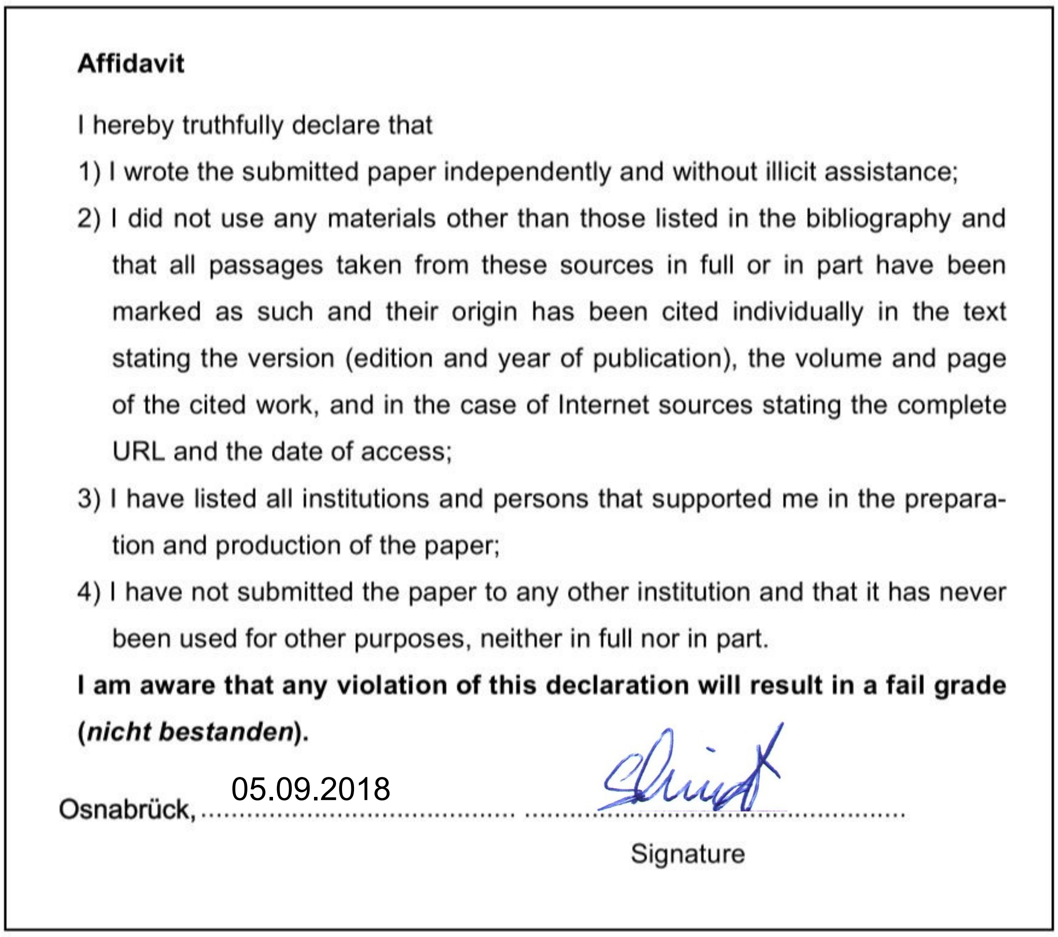
Rodriguez-Esteban, Raul. 2009. “Biomedical Text Mining and Its Applications.” Edited by Fran Lewitter. *PLoS Computational Biology* 5 (12): e1000597. https://doi.org/10.1371/journal.pcbi.1000597.

Simpson, Matthew S., and Dina Demner-Fushman. 2012. “Biomedical Text Mining: A Survey of Recent Progress.” In *Mining Text Data*, edited by Charu C. Aggarwal and ChengXiang Zhai, 465–517. Boston, MA: Springer US. https://doi.org/10.1007/978-1-4614-3223-4_14.

Zhou, Li, Joseph M. Plasek, Lisa M. Mahoney, Neelima Karipineni, Frank Chang, Xuemin Yan, Fenny Chang, Dana Dimaggio, Debora S. Goldman, and Roberto A. Rocha. 2011. “Using Medical Text Extraction, Reasoning and Mapping System (MTERMS) to Process Medication Information in Outpatient Clinical Notes.” In *AMIA Annual Symposium Proceedings*, 2011:1639. American Medical Informatics Association.

5.2 Websites

BioNLP-ST 2013. n.d. Accessed August 31, 2018.
URL: http://2013.bionlp-st.org/Intro.
Introduction to the Linguistic String Project. n.d. Accessed August 31, 2018.
URL: https://cs.nyu.edu/cs/projects/lsp/LSP_intro.html.
Python Docs Sequence Matcher. n.d. Accessed September 5, 2018.
URL: https://docs.python.org/2.4/lib/sequencematcher-examples.html.
Yearly Citation Totals from 2017 MEDLINE/PubMed Baseline. n.d. U.S. National Library of Medicine.
URL: https://www.nlm.nih.gov/bsd/licensee/2017_stats/2017_Totals.html.



Affidavit

I hereby truthfully declare that

1) I wrote the submitted paper independently and without illicit assistance;
2) I did not use any materials other than those listed in the bibliography and that all passages taken from these sources in full or in part have been marked as such and their origin has been cited individually in the text stating the version (edition and year of publication), the volume and page of the cited work, and in the case of Internet sources stating the complete URL and the date of access;
3) I have listed all institutions and persons that supported me in the preparation and production of the paper;
4) I have not submitted the paper to any other institution and that it has never been used for other purposes, neither in full nor in part.

I am aware that any violation of this declaration will result in a fail grade (*nicht bestanden*).

Osnabrück, 05.09.2018 .... [illegible]

Signature