Building a Relation Extraction Baseline for Gene-Disease Associations: A Reproducibility Study

Laura Menotti[1]

[1]Department of Information Engineering, University of Padua, Padua, Italy

Abstract

Reproducibility is an important task in scientific research. It is crucial for researchers to compare newly developed systems with the state-of-the-art to assess whether they made a breakthrough. However previous works may not be immediately reproducible, for example due to the lack of source code. In this work we reproduce DEXTER, a system to automatically extract Gene-Disease Associations (GDAs) from biomedical abstracts [1]. The goal is to provide a benchmark for future works regarding Relation Extraction (RE), enabling researchers to test and compare their results.

Keywords

Relation Extraction, Reproducibility, Gene-Disease Associations

1. Introduction

Biomedical literature is a rich source of information on Gene-Disease Associations (GDAs) that could help physicians in assessing clinical decisions and improve patient care. GDAs are publicly available in databases containing relationships between gene/miRNA expression and related diseases such as specific types of cancer. Most of these resources, such as DisGeNET [2, 3], miR2Disease [4] and BioXpress [5, 6], include also manually curated data from publications. Human annotations are expensive and cannot scale to the huge amount of data available in scientific literature (e.g., biomedical abstracts). Therefore, developing automated tools to identify GDAs is getting traction in the community [7]. Such systems employ Relation Extraction (RE) techniques to extract information on gene/microRNA expression in diseases from text. Once an automated text-mining tool has been developed, it can be tested on human annotated data or it can be compared to state-of-the-art systems.

In this context, the state of the art is DEXTER, a rule-based system that extracts gene/microRNA expressions in diseases from biomedical abstracts [1]. Unfortunately, DEXTER's source code is not publicly available hence researchers rely only on data provided by the authors to evaluate newly developed systems. In this work we reproduce DEXTER to provide a benchmark for RE, thus enabling researchers to test and compare their results to a state-of-the-art baseline.

The rest of the paper is structured as follows: Section 2 introduces DEXTER and its main components, Section 3 describes our implementation of DEXTER by highlighting the main differences and commonalities with the original system, Section 4 presents the obtained results and Section 5 draws some conclusions.

FDIA 2022: Symposium on Future Directions in Information Access (FDIA), July 07-20, 2022, Lisbon, Portugal.

laura.menotti@studenti.unipd.it (L. Menotti)

The implemented version of DEXTER is available in the following git repository: https://github.com/mntlra/DEXTER.

2. Original System

DEXTER is a system that extract Gene-Disease Associations (GDAs) starting from biomedical abstracts [1]. DEXTER has been published in *Database: The Journal of Biological Databases and Curation* and has attracted 11 citations so far. To the best of our knowledge it is the most recent system for RE on Gene-Cancer Associations. One of the main objectives of the original work is to create a tool to automatically extend expression databases like BioXpress [5, 6], that contains gene/miRNA expression associated with cancer. The authors focus on a specific set of sentences that report differential expression of genes with or without an explicit comparison between tissues or disease states. Such choice can be motivated by the additional constraints needed to extend BioXpress, i.e. the detection of differential expression of genes compared to healthy tissues. This kind of sentences usually fit into two broad categories called TypeA and TypeB sentences. In TypeA sentences a gene's expression is typically contrasted between two different samples or conditions. Such sentences are typically comparative as shown in Example 1, where *Nucleolin expression* is analyzed in non-small cell lung carcinoma (NSCLC) tissues compared with normal lung tissues.

> Example 1: Nucleolin expression was higher in NSCLC tissues than adjacent normal lung tissues. *(TypeA sentence)*

On the other hand, TypeB sentences still provide the expression level of a gene in a particular sample, however there is no explicit comparison. We can see an example of this type of sentences in Example 2, where *miR-630* is detected in NSCLC tissuses without any comparison with other samples.

> Example 2: Our results showed that miR-630 was significantly down-regulated in NSCLC tissues and cell lines. *(TypeB sentence)*

Among biomedical literature regarding gene expression in diseases one can also find a third category called TypeC sentences. In such sentences gene's expression level and related diseases are still reported, however there is no explicit association between the gene and the disease.

> Example 3: Over-expression of C1GALT1 enhanced breast cancer cell growth, migration and invasion *in vitro* as well as tumor growth *in vivo. (TypeC sentence)*

In Example 3, we see that *when* C1GALT1 is over-expressed in breast cancer, cell-growth is enhanced. However, there is no explicit association between the gene and breast cancer, meaning there is no information about whether C1GALT1 is typically over-expressed in breast cancer or not. DEXTER does not extract information from such sentences.

DEXTER is an expert system, whose rules are based on the syntactic structure of TypeA and TypeB sentences and it extracts information on gene or miRNA expressed in an associated disease. The system is developed mainly in Python and Java and it is based on several modules as shown in Figure 1, each dealing with a different part of the computation.

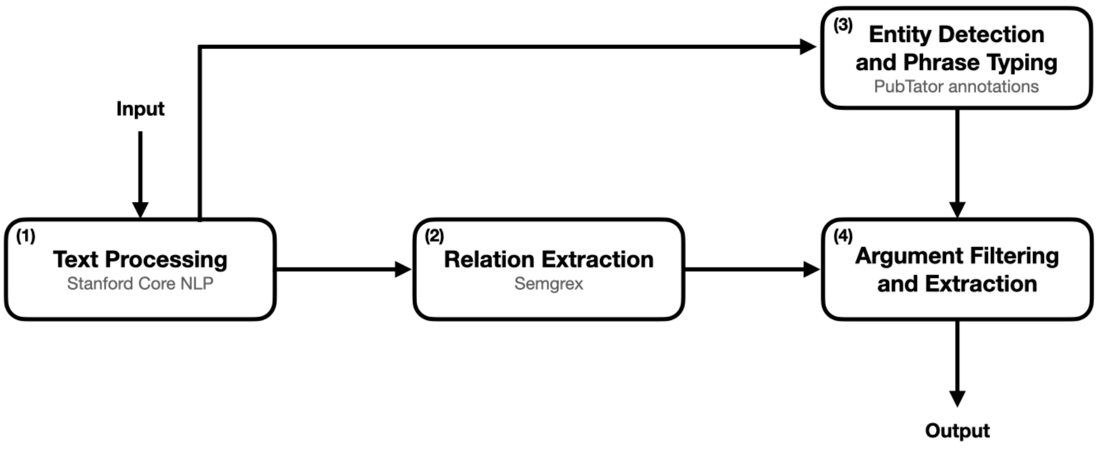

Figure 1: DEXTER modules defined in the original paper. Arrows indicates the direction of the computation, providing an overview of the input of each block.

In the text processing block, a given medical abstract is tokenized and split into sentences. Such operation is performed by Stanford CoreNLP toolkit, which is a pipeline that takes raw text as input and returns a set of Natural Language Processing (NLP) annotations as output [8]. CoreNLP can be useful in a number of NLP tasks including Part-Of-Speech tagging, Named Entity Recognition and Dependency Parsing. The RE module extracts information from the sentences by using patterns based on lemmas, part-of-speech tags and dependency labels. Therefore using this library is essential for achieving DEXTER's objectives. After tokenization and sentence splitting, sentences that do not contain predefined trigger words are discarded and not processed any further. Such terms are available in one of the two supplementary files provided as appendixes of the original paper and they are used to detect sentences that might contain expressions and comparative relations. Each sentence is parsed using the Charniak-Johnson parser [9, 10] with David McClosky's adaptation to the biomedical domain [11] because they provide constituency parse trees that are then converted into Standard Dependency Graphs (SDGs) [12]. The resulting SDG is further processed collapsing and propagating dependencies to properly treat sentences involving conjunctions.

In the RE module, patterns are written in Semgrex[1], which is also part of the Stanford NLP toolkit and it allows the matching of nodes and edges in a dependency graph. In particular, nodes in a SDG are the tokens corresponding to sentence words and patterns are written to check a set of attributes for each token. Attributes comprise lemmas, part-of-speech tags and dependency labels between the considered node and already-matched nodes. Authors rely on a previous work where they identify comparative structure in biomedical text [13] to write comparison patterns used to extract the components. All comparison patterns are available in a supplementary file of the original paper. In typeA sentences, the RE module extracts the aspect being expressed (Compared Aspect), the level of expression of such aspect (Scale Indicator) and the entities being compared in the sentence (Compared Entities), that can be two samples of tissue or two disease states. In TypeB sentences the components extracted are basically the

[1]https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/semgraph/semgrex/SemgrexPattern.html

Table 1
Components extracted from TypeA and TypeB sentences from Example 4. The Compared Aspect is the gene/miRNA being expressed, the Scale Indicator is the level of expression of such aspect and the Compared Entities are the entities being compared in the sentence. Note that in TypeB sentences there is only one entity since the structure of the phrase contain only one tissue or disease state where the gene/miRNA has been detected.

	Sentence Type	Scale Indicator	Compared Aspect	Compared Entity 1	Compared Entity 2
#1	TypeA	Elevated	Sam68	NSCLC tissues	non-cancerous tissues
#2	TypeA	Decreased	Lynx1	lung cancers	normal lung
#3	TypeB	Increased	EphA2	NSCLC metastases	
#4	TypeB	Lower	miR-195	tumor tissues	

same, the only difference is that there is only one entity where the gene/miRNA is expressed. A different terminology is used in the original paper to identify TypeB components such as Expressed Aspect, Expressed Location and Level Indicator. To better understand how this module works we provide some examples both for TypeA and TypeB sentences in Example 4 and the corresponding components extracted by the RE module are shown in Table 1. For the sake of simplicity, we used the same terminology both for TypeA and TypeB components.

Example 4:

#1 The expression of *Sam68* was significantly *elevated* in *NSCLC tissues* as compared with *adjacent non-cancerous tissues.*[PMID:24522888]

#2 *Lynx1* levels are *decreased* in *lung cancers* compared to *adjacent normal lung.*[PMID:26025503]

#3 Expression of *EphA2* is *increased* in *NSCLC metastases.*[PMID:20360610]

#4 We demonstrated that *miR-195* expression was *lower* in *tumor tissues* and was associated with poor survival outcome.[PMID:25840419]

In Example 4, the first two sentences are of TypeA since there is an explicit comparison between two entities. In particular, in the first sentence *'Sam68'* expression is compared in *'non-small-cell lung carcinoma (NSCLC)'* and *'non-cancerous tissues'*. Consider sentence #2, the RE module extracted *'Linx1'* as the gene expressed (i.e., the Compared Aspect), *'decreased'* as the level of expression (i.e., the Scale Indicator) and the entities being compared (i.e., Compared Entity 1 and 2) are *'lung cancers'* and *'adjacent normal lung'* as shown in Table 1. On the other hand, sentences #3 and #4 are of TypeB since there is only one entity where the gene/miRNA is expressed. In fact, consider sentence #3, increased expression of *'EphA2'* is detected in *'NSCLC metastasis'* but there is no other entity compared to it. In this case we can see from Table 1 that the RE module extracted *'EphA2'* as the Compared Aspect, *'increased'* as the Scale Indicator and there is only one Compared Entity identified in *'NSCLC metastases'*.

The entity detection and phrase typing module takes sentences parsed by the text processing module and determine if there are terms referring to entities of type gene/miRNA, expression or disease/disease-sample. PubTator [14] is an automatic system providing annotations of biomedical concepts such as genes and diseases whose pre-computed annotations are employed to detect genes and diseases mentions in abstracts. On the other hand, microRNA mentions are detected using regular expressions based on the naming convention described in miRBase [15]. Finally, to determine whether a phrase is of type ’Expression’ its terms are checked against a list of expression triggers such as ’expression’, ’level, ’over-expression’, etc. The same strategy has been adopted to detect disease-sample phrase type. In this case triggers can be ’tissues’, ’cells’, ’patients’, ’samples’, etc. A full list of such triggers is provided in the supplementary files of the original paper.

The argument filtering and extraction module consists of two main steps: verify if the tokens matched by the RE module are of the right type, i.e. contains the required mentions, and extract the relevant information such as the gene/miRNA expressed and the associated disease. To perform both tasks this module takes the results of the RE module and the mentions identified in the previous block as input. For TypeA sentences, the compared aspect must be of type expression, while the two compared entities need to be of type disease/disease-sample. Similarly, for typeB sentences the expressed aspect and the expression level must be respectively of type expression and disease/disease-sample. If the checks succeed, relevant information are extracted from the results computed by the RE module. This includes the expressed gene, its level and the associated disease. The gene/miRNA expressed is extracted from the compared aspect/expressed aspect argument using the mentions extracted by the entity detection and phrase typing module. If the system detects a gene expression it also returns the corresponding NCBI Gene ID downloaded from PubTator together with the gene mentions. Expression level information is captured by the RE module in the scale indicator/level indicator argument, therefore the level is normalized to high or low by matching the argument against a list of triggers available in the supplementary files. Some examples of triggers for ’high’ expression are ’over-expressed’, ’increased’, etc, while terms like ’low’, ’decreased’, ’down-regulated’ refers to ’low’ expression level. Extracting the associated disease requires some extra care. Disease mentions downloaded from PubTator are searched in the compared entities/expression location components, however in some cases the entities contain only generic diseases such as ’tumor’, ’cancer’ or ’disease’. In these cases, the associated disease is extracted from the abstract by looking for disease mentions in the title or in the first sentence. Alternatively the associated disease can be inferred from sentences in the ’methods’ part of the abstract or from sentences describing the experimental set-up. Such sentences are detected by checking if they contain certain ’investigation triggers’ such as ’investigated’, ’examined’, ’analyzed’, etc or ’analyzed triggers’ such as ’tested’, ’collected’, ’explored’, etc. PubTator returns disease mentions normalized to MEDIC IDs, however such IDs are mapped to DOIDs to allow an easy integration in BioXpress.

3. Implemented Version

The original work does not share the source code, therefore for the reproducibility we used the description of the paper and the supplementary files containing comparison patterns and trigger lists. As mentioned before, DEXTER is developed using both Python and Java therefore each block run separately from the others. Our goal is to implement an end-to-end system publicly available and easy to reproduce without knowing exactly what each block does. For this reason, we decided to implement DEXTER as an end-to-end application that takes as input biomedical

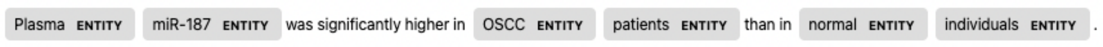

Figure 2: Standard SpaCy pipeline retrieved 6 entities for sentence in Example 5. Entities detected by the standard pipeline are highlighted in grey. The visualization is done using the DisplaCy package.

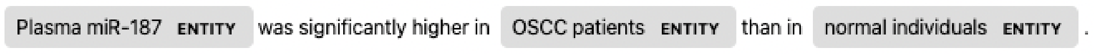

Figure 3: Our custom entity expansion component detected 3 entities for sentence in Example 5, which corresponds to the Compared Aspect and the two Compared Entities. Entities detected by the standard pipeline are highlighted in grey. We can notice how the custom component collapsed the 6 entities detected by the standard pipeline into 3 entities. The visualization is done using the DisplaCy package.

abstracts and returns relevant information on the expressed gene/microRNA and the associated disease. We preserved the overall block architecture and we made some changes in each block to enable a seamless integration of the different modules. The system is entirely developed in Python therefore it was no longer possible to use Stanford CoreNLP toolkit since it is written in Java. CoreNLP developers published Stanza [16], that is a Python NLP package that includes an interface to the CoreNLP Java package, however we decided not to use it since at the time of writing it does not offer support for dependency parsing. Instead we used the SpaCy library [17], which provides annotated text using some trained pipelines available on their website.[2] We used the `en_sci_sm` trained pipeline provided by ScispaCy [3], that is a Python package containing spaCy models for processing biomedical data. In addition, we added a custom component to the SpaCy pipeline to expand entity mentions using hand-craft rules which will be useful when extracting information from the matched components. An example of how entity expansion works is shown in Figures 2 and 3 for the sentence in Example 5.

> Example 5: Plasma miR-187 was significantly higher in OSCC patients than in normal individuals.

In particular, we can notice from Figure 2 and 3 how *OSCC patients* were considered as two separate entities but after the entity expansion they form a single entity.

Once the system has parsed each sentence we check the presence of miRNA mentions. In that case we tokenize again the sentence to make sure each miRNA mention is considered as a single token. Such operation ensures dependencies are computed correctly and it will be useful in the argument filtering and extraction module.

The RE module is very similar to the one described in the paper [1] since the Dependency-Matcher library from SpaCy allows to match patterns within the dependency tree of a sentence using Semgrex operators. As a consequence we adapted the comparison patterns available in the supplementary files to match the patterns suitable for the DependencyMatcher library and we added some rules to return matches for more sentences. In fact, we noticed that SpaCy returns different dependency trees than the Stanford CoreNLP library. For this reason we adapted the rules to better handle these cases and to prevent any foreseeable loss of information.

[2]https://spacy.io/usage/models
[3]https://allenai.github.io/scispacy/

The entity detection and phrase typing module and the argument filtering and extraction module are not significantly different from the original version of DEXTER. Since we downloaded the PubTator annotations in batches we store all the annotations together in such a way that if PubTator failed to detect some mentions from a specific PubMed ID (PMID) we can still retrieve such information if the same gene or disease appears in other abstracts. Furthermore, we added some term triggers for the expression level normalization to cover all possible levels that appear in the sentences.

4. Evaluation

Authors tested DEXTER on data relevant to BioXpress and on a large-scale dataset. To extend the literature-based portion of the BioXpress database they considered three use-cases. Firstly, they evaluated DEXTER on a set of abstracts related to lung cancer. Secondly, they focused on a set of abstracts related to a set of genes, called glycosyltransferases (GTs), which are a set of enzymes. Lastly, authors run the system on a set of abstracts related to the role of microRNA in cancer. The authors tested the system against human annotated data and published the results computed by DEXTER on the website of BioXpress.[4]

To evaluate the effectiveness of the reproduced version of DEXTER we run our implementation of the system on the sentences of the three use-cases available in BioXpress and compared the results in terms of correct sentence type and gene expression level. This means we consider our results correct when we extract the same gene and expression level as in the input data and when we correctly classify sentences as TypeA and TypeB with respect to the input data as well.

Overall, our implementation correctly parse the 97% of the sentences, these are sentences for which we are able to extract GDAs and that do not raise any exceptions during computation. Our system performance is shown in Table 2. Results refer to all three use-cases and we consider the data provided in BioXpress as ground truth. The first two columns show the performance metrics when we run our implementation using the full DEXTER pipeline, i.e. using the PubTator annotations. The other two columns report the system performance when using the annotations provided by the input data instead of the PubTator annotations. As already mentioned, metrics are computed in terms of correct gene expression level and correct sentence type. The results show that the system has been reproduced to a reasonable degree since the performance drop is negligible and mostly due to the use of SpaCy rather than the system implementation itself. In fact, after a deeper analysis we noticed several parse trees computed by SpaCy were different from the one returned by the CoreNLP library therefore most unparsed sentences can be attributed to it. Additionally, it is important to notice that PubTator is an automated tool based on a neural model that is periodically retrained therefore the version of PubTator we use to retrieve mentions is different from the one used in the original paper to compute the datasets we downloaded from BioXpress. As a consequence, missing mentions affect negatively the accuracy performance of the system as shown in Table 2. In particular, expression level accuracy is 84% when using PubTator mentions however if we use the mentions that have been already extracted by the original approach – i.e., those using a different PubTator version – the overall performance improves, especially as far as expression level is concerned

[4]https://hive.biochemistry.gwu.edu/bioxpress

Table 2

Performance metrics. Results refer to all three use-cases and we consider the data provided in BioXpress as ground truth. The first two columns show the performance metrics when we run our implementation using the full DEXTER pipeline, i.e. using the PubTator annotations. The other two columns report the system performance when using the annotations provided by the input data instead of the PubTator annotations. Metrics are computed in terms of correct gene expression level and correct sentence type.

	Full Pipeline		Without PubTator Annotations	
	Expression Level	Sentence Type	Expression Level	Sentence Type
Accuracy	0.8436	0.9353	0.9293	0.9326
Precision	0.8436	0.9377	0.9296	0.9349
Recall	0.8436	0.9352	0.9293	0.9326
F1 score	0.8436	0.9359	0.9294	0.9332

where accuracy raise to 93%. This means that most of the wrongly detected levels are due to the different version of PubTator and do not depend on the system implementation itself. Nevertheless, the dependency on PubTator limits the reproducibility of DEXTER and opens to some potential issues to apply it to other documents.

5. Conclusions

This work reproduces DEXTER, a system to automatically extract GDAs from biomedical abstracts [1]. The goal is to provide a baseline for future works regarding RE. The system implementation parsed 97% of the input sentences while discarded sentences are mostly due to missing PubTator annotations or problems related to Dependency Parsing. The system achieved an accuracy of 84% on the gene expression level. Such value raises up to 93% if input mentions are used instead of PubTator annotations. Results in Table 2 demonstrate that the system is reproducible and it can be used as a benchmark for future works. The implemented version of DEXTER is available on GitHub. In this way researchers can compare newly developed systems with a state-of-the-art instead of merely rely on annotated data.

In conclusion, implementing this system highlighted its limits. DEXTER is a rule-based model hence information are extracted based on patterns written to match sentences with a very specific syntactic structure, namely TypeA and TypeB sentences. If biomedical abstracts do not follow such structure, as for TypeC sentences, they are discarded. This is a huge limitation since sentences usually have a wide variety of syntactic forms and it could cause undesired loss of information.

Acknowledgments

This work was supported by the ExaMode Project, as a part of the European Union Horizon 2020 Program under grant 825292. We also sincerely thank Gianmaria Silvello and Stefano Marchesin whose suggestions helped in improving this study.

References

[1] S. Gupta, H. Dingerdissen, K. E. Ross, Y. Hu, C. H. Wu, R. Mazumder, K. Vijay-Shanker, DEXTER: Disease-Expression Relation Extraction from Text, Database 2018 (2018). URL: https://doi.org/10.1093/database/bay045. doi:10.1093/database/bay045, bay045.

[2] J. Piñero, N. Queralt-Rosinach, A. Bravo, J. Deu-Pons, A. Bauer-Mehren, M. Baron, F. Sanz, L. I. Furlong, Disgenet: A discovery platform for the dynamical exploration of human diseases and their genes, Database 2015 (2015). doi:10.1093/database/bav028.

[3] A. Bauer-Mehren, M. Rautschka, F. Sanz, L. I. Furlong, Disgenet: A cytoscape plugin to visualize, integrate, search and analyze gene-disease networks, Bioinformatics (Oxford, England) 26 (2010) 2924–6. doi:10.1093/bioinformatics/btq538.

[4] Q. Jiang, Y. Wang, Y. Hao, L. Juan, M. Teng, X. Zhang, M. Li, G. Wang, Y. Liu, mir2disease: a manually curated database for microrna deregulation in human disease, Nucleic acids research 37 (2008) D98–104. doi:10.1093/nar/gkn714.

[5] H. Dingerdissen, J. Torcivia-Rodriguez, T.-C. Chang, R. Mazumder, R. Kahsay, Biomuta and bioxpress: Mutation and expression knowledgebases for cancer biomarker discovery, Nucleic Acids Research 46 (2018) D1128–D1136. doi:10.1093/nar/gkx907.

[6] Q. Wan, H. Dingerdissen, Y. Fan, N. Gulzar, Y. Pan, T.-J. Wu, C. Yan, H. Zhang, R. Mazumder, Bioxpress: An integrated rna-seq-derived gene expression database for pan-cancer analysis, Database : the journal of biological databases and curation 2015 (2015). doi:10.1093/database/bav019.

[7] S. Marchesin, G. Silvello, TBGA: a large-scale gene-disease association dataset for biomedical relation extraction, BMC Bioinform. 23 (2022) 111. URL: https://doi.org/10.1186/s12859-022-04646-6. doi:10.1186/s12859-022-04646-6.

[8] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, D. McClosky, The Stanford CoreNLP natural language processing toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 55–60. URL: https://aclanthology.org/P14-5010. doi:10.3115/v1/P14-5010.

[9] E. Charniak, A maximum-entropy-inspired parser, in: 1st Meeting of the North American Chapter of the Association for Computational Linguistics, 2000. URL: https://aclanthology.org/A00-2018.

[10] E. Charniak, M. Johnson, Coarse-to-fine n-best parsing and MaxEnt discriminative reranking, in: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Association for Computational Linguistics, Ann Arbor, Michigan, 2005, pp. 173–180. URL: https://aclanthology.org/P05-1022. doi:10.3115/1219840.1219862.

[11] D. McClosky, E. Charniak, M. Johnson, Automatic domain adaptation for parsing, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT '10, Association for Computational Linguistics, USA, 2010, p. 28–36.

[12] M.-C. de Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, C. D. Manning, Universal Stanford dependencies: A cross-linguistic typology, in: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), European Language Resources Association (ELRA), Reykjavik, Iceland, 2014, pp. 4585–4592. URL:

http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf.

[13] S. Gupta, A. A. Mahmood, K. Ross, C. Wu, K. Vijay-Shanker, Identifying comparative structures in biomedical text, in: BioNLP 2017, Association for Computational Linguistics, Vancouver, Canada„ 2017, pp. 206–215. URL: https://aclanthology.org/W17-2326. doi:10.18653/v1/W17-2326.

[14] C.-H. Wei, A. Allot, R. Leaman, Z. Lu, PubTator central: automated concept annotation for biomedical full text articles, Nucleic Acids Research 47 (2019) W587–W593. URL: https://doi.org/10.1093/nar/gkz389. doi:10.1093/nar/gkz389.

[15] S. Griffiths-Jones, R. Grocock, S. Dongen, A. Bateman, A. Enright, mirbase: microrna sequences, targets and gene nomenclature, Nucleic acids research 34 (2006) D140–4. doi:10.1093/nar/gkj112.

[16] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, C. D. Manning, Stanza: A Python natural language processing toolkit for many human languages, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020. URL: https://nlp.stanford.edu/pubs/qi2020stanza.pdf.

[17] M. Honnibal, M. Johnson, An improved non-monotonic transition system for dependency parsing, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1373–1378. URL: https://aclweb.org/anthology/D/D15/D15-1162.