# End-to-end Acronym Expander System for Bulgarian language

Monthly Report 3 for MSc IS

Iliya Georgiev
University of Amsterdam
Amsterdam, The Netherlands
iliya.georgiev@student.uva.nl

## ABSTRACT

Scientists from different fields such as biology, chemistry, and physics are using acronyms to make their communication faster. The use of acronyms saves time and space, but it makes the text more complex. Not everyone is an expert and has knowledge in these fields, therefore a system for acronym expansion is needed. This system would identify acronyms and find their expansion, so one can understand the meaning of the acronym with ease. An acronym expansion (AE) can be found in the text in a predefined database or can be extracted from the internet. This thesis will focus on the AE problem within the Bulgarian language. The report will provide an extension to an existing end-to-end system for AE to support the Bulgarian language and add support for acronym construction without using parentheses. Moreover, a new data set will be developed and it will be used to test the system. Finally, the author will consider the case where multilingual acronyms are used within the text. The system will be evaluated by comparing a baseline, the current system - end-to-end system for acronym expansion, and the new system based on the precision, recall, F1-score of the new data set, and the execution time it takes to process it.

**Main supervior UVA:** João Lebre Magalhães Pereira
(j.p.pereira@uva.nl)

## 1 METHODOLOGY AND EXPERIMENTAL SETUP

In this section, the author will discuss the methodology and the experimental setup of this project.

### 1.1 Data collection

The data will be collected from Wikipedia - Bulgaria[1], as well as different online newspapers and governmental websites. The extraction of the articles will be done within 10 generic categories - Psychology, Biomedical, Sports, Politics, Financial, Engineering, Computer science, Math, Education, and Biology. The author will manually extract at least 20 documents per topic and will save these documents as *.txt* files on the server using the WikiExtractor [2]. The articles will be randomly selected without taking any requirements into account. These documents will be later annotated by the author and external readers using the Acronym Expander Annotation System.

### 1.2 Data preparation - Acronym Expander Annotation System

To be able to use the gathered data for this research, it needs to be pre-processed. This process includes text pre-processing of the Wikipedia data and merging multiple documents to a data set. A custom annotation system was built by the author and other participators in the project to simplify the annotation process. The system is built upon Python[3] using Flask[4] as a web framework. It uses the Flask Templates to serve the front-end to the users. The front-end of the system is built upon HTML, CSS and JavaScript, using Bootstrap as a front-end framework. The system is deployed on a Google Cloud server and it accessible by annotators at https://annotation-acx.nw.r.appspot.com/.

This system allows external readers and the author to annotate the collected data. This is done by providing the user with an extracted text and asking him/her to match all acronyms within the text with the right expansion if there is one. In case the expansion is not found in the text, the user will have to annotate the acronym and its expansion according to the context of the text. Moreover, the annotators should provide the language of all of the acronyms that they found. The system will give specific instruction on how the annotation should be performed. After the annotation is done, the annotated acronyms and their expansions will be stored in the database. The system keeps track of all of the articles that are annotated by the user to make sure that a user will not annotate the same article twice. Moreover, since each of the articles should be annotated by two different readers, the system prioritizes articles that have not been an annotated yet or the ones that have been annotated only one time. Furthermore, the system uses a smart article lock process, which assigns a document to annotator for 5 hours. If the annotator does not finish the task at this time, then the system will assign this document to another user to prevent articles from being not annotated due to lack of user activity.

After the annotation is performed, then the text of the annotated document with the annotated acronyms and their expansions will be stored on the server. This collection will be then merged and converted to a data set used to test the system. The data will be distributed into 3 different data sets to be used for the three different parts of the system - Acronym Identification(AI), Acronym Disambiguation(AD), and Acronym Expansion(AE). The AI data set will compose of more articles within text expansion, while the AD data set will mostly use articles where the expansion is not inside the text. For the AE, all of the articles will be used for a test data set.

[1] https://bg.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D0%BD%D0%B0_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0
[2] https://github.com/attardi/wikiextractor
[3] https://python.org
[4] https://flask.palletsprojects.com/

## 1.3 Methodology

Currently, the end-to-end system recognizes expansions of the acronyms, only if the acronyms are in parentheses. For example, the pair Personal Identification Number (PIN) will be recognized, however, the pair, Personal Identification Number - PIN will not be recognized. The addition to the current implementation will be to add support for the Bulgarian language and add acronym recognition when the acronym is not in parentheses in the first encounter. At the moment, the system is divided into two parts to effectively solve the acronym expansion problem.

*1.3.1 Acronym Identification.* For Acronym Identification (AI), two strategies will be tested and observed which one will perform better in the Bulgarian language. The first one will be using a rule-based approach, implementing the Schwartz et al. [2] algorithm and the MadDog version of this algorithm [3]. The second strategy used for the implementation of the Bulgarian language will be the machine learning approach using SciBERT models[1]. Moreover, to recognize the acronym when no parentheses are used new AI strategy will be added. This strategy will recognize other patterns for acronyms that do not include parentheses, such as an expansion followed by an acronym inside the parentheses ("-"). These patterns will identify acronyms and expansions that use the following structure, EXPANSION - ACRONYM or ACRONYM - EXPANSION. For example, the current AI techniques - MadDog-In [15] and Schwartz [12] used in the existing system will not recognize the following acronym-expansion pair, Natural language processing - NLP, because the acronym is not within parentheses - Natural language processing (NLP). To recognize acronyms and expansions outside of the parenthesis the author will implement and test the rule-based AI techniques proposed by Yarygina et al. [16] using regular expressions and pattern matching approach introduced by Mohammed et al. [11]. This new strategy will not be dependent on the language and therefore it will be available for all languages that are implemented in the system.

*1.3.2 Acronym Disambiguation.* For the Acronym Disambiguation (AD), the system has to look up the acronym in an external database containing documents and the pairs of acronyms and expansions that can be found in these documents. After this, the system decides based on the context which expansion fits best. To do this, the author will try different techniques such as:

- Representator - this technique will use Term Frequency-Inverse Document Frequency (TF-IDF) to calculate the frequency of the acronym usage in the documents, Latent Dirichlet Allocation (LDA) to determine the topic of the document, and Doc2Vec to discover how similar the documents are.
- Classification - approaching the problem with ML, these techniques will include SVM, Logistic Regression (LR), and Random Forests (RF).
- Combination - combining different techniques might increase the accuracy of the system. This is why the author will use a different combination of techniques and evaluate which one fits best for the Bulgarian language.

## 1.4 Experimental setup

The existing system runs on Python 3.7 and it can be installed on any Unix-based operating system. To install the system you should follow the steps that can be found on https://web.tecnico.ulisboa.pt/~ist164790/acx/.

## 1.5 Evaluation of the results

To evaluate the results of this study, the author will separate the system into three parts - acronym identification, acronym disambiguation, and acronym expansion. As a base benchmark, this study will use a random approach, where the disambiguation of the acronym is randomly selected from the possible list of candidates. Moreover, the system which is already available in English will also be used as a benchmark, comparing the techniques discussed in section 3.3. The evaluation will be using the newly created data sets in Bulgarian to assess the performance of the system when used in Bulgarian. The author will use the following performance metrics:

- *Precision* - number of correctly retrieved acronym-expansion pairs divided by the total number of retrieved acronym-expansion pairs.
- *Recall* - number of correctly retrieved acronym-expansion pairs divided by the total number of acronym-expansion pairs in the data set.
- *F1-score* - harmonic mean of the precision and recall.

$$F_1 = 2 \cdot \left( \frac{precision \cdot recall}{precision + recall} \right)$$

- *Execution time* - related to the amount of time needed for the system to process the data set and not the amount of time needed to train the system

These metrics will be used to evaluate the performance of the system when using previously annotated data sets.

Moreover, as mentioned earlier the thesis will focus on handling multilingual disambiguation, or the use of more than one language when defining acronyms in the document. To evaluate what is the impact of this, the author will look into the measures mentioned above when multilingual acronyms are used. It will be investigated if the system performance is worse when the text contains multilingual acronym expansion pairs, for example, "Интерфейс за малки компютърни системи (SCSI)".

## REFERENCES

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. 2020. SCIBERT: A pretrained language model for scientific text. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020), 3615–3620. https://doi.org/10.18653/v1/d19-1371 arXiv:1903.10676

[2] Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* (2003), 451–462. https://doi.org/10.1142/9789812776303_0042

[3] Amir Pouran Ben Veyseh, Franck Dernoncourt, Walter Chang, and Thien Huu Nguyen. 2021. MadDog: A web-based system for acronym identification and disambiguation. *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the System Demonstrations* (2021), 160–167. https://doi.org/10.18653/v1/2021.eacl-demos.20 arXiv:2101.09893