# Acronym Expansion System in Bulgarian

## Introducing a new rule-based algorithm that identifies acronyms without parentheses

submitted in partial fulfillment for the degree of master of science

Iliya Georgiev
13399640

Master Information Studies
information systems
Faculty of Science
University of Amsterdam

Submitted on 30.06.2023

| | **Daily Supervisor** | **Examiner** |
|---|---|---|
| **Title, Name** | João Lebre Magalhães Pereira | Prof. Dr. P.T. Groth |
| **Affiliation** | University of Amsterdam | University of Amsterdam |
| **Email** | j.p.pereira@uva.nl | p.t.groth@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 focuses on the AE problem within the Bulgarian language and proposes a new no-parentheses algorithm for Acronym Identification (AI). Moreover, a new data set is developed and it is used to test the system. The paper suggests that the newly introduced algorithm significantly increases the recall and F1 Measure in all of the data sets in the system. However, one should consider testing the system with a richer Bulgarian data set and introducing different languages to the system to possibly find a correlation between the acronyms expressed in multiple languages and alphabets.

## KEYWORDS

acronym expansion, Bulgarian data set, acronym identification without parenthesis, acronym identification, annotation system

## GITHUB REPOSITORY

https://github.com/isgeorgiev/uva-thesis

## 1 INTRODUCTION

In recent years, the number of acronyms used in both the scientific and nonscientific domains has drastically increased [2]. Acronyms (or abbreviations) are a short form of longer phrases, usually constructed from the first letter of each word in the phrase. The reasons why people are using acronyms are mainly to reduce the time needed to express themselves and space while writing news articles, scientific papers, and posts on social media.

Whenever people are reading through a document and there is an acronym used in it, they immediately look for the meaning of this acronym. Usually, after the first appearance of the acronym in the text, it is common that the expansion in parentheses follows immediately after. However, when those acronyms are generally accepted in the field, often their expansion is omitted [2]. This usage of abbreviations raises the problem of acronym expansion (AE). If the acronym expansion is not available in the text, one should understand the context of the text to determine which expansion the author intended to use within the text. Experts in different fields are introducing abbreviations that are applicable for their domain, however, those are not always globally unique. For example, the acronym 'DM' means 'Diabetes Mellitus' in the medical domain and 'Direct Message' in the technical domain.

The above-mentioned problem can be solved by building a system that considers the following two steps: **Acronym Identification (AI)** - find the acronym and its expansion in the text and **Acronym Disambiguation (AD)** - find the correct expansion that corresponds to the acronym. Over the past two decades, there has been significant research towards finding a solution for those two steps [13, 17–19, 22]. The limitation of these studies is that they focus only on identifying acronyms that are nested between special characters, such as parenthesis [17, 19]. This study will focus on exploring possibilities for acronym identification that does not include parenthesis. Moreover, the previous studies focus mainly on studying Latin alphabet languages such as English, Spanish, German and etc. Since the Bulgarian language (BL) is using the Cyrillic alphabet and not Latin, it is difficult for such systems to perform AI and AD effectively. A natural processing language model that accepts the BL is needed for the system to support Bulgarian documents more effectively[6].

This thesis aims to contribute to an existing end-to-end system for acronym expansion. The system currently operates in English, and it is developed by Pereira et al. [10], who is also the supervisor of this project. The system takes a document as input and it outputs the same document with all of its acronyms expanded. The existing solution reaches its peak performance while using MadDog-in [22] for AI and Support Vector Machine (SVM) [7] and Doc2Vec [14] for AD. This thesis implements new AI techniques in the system. Apart from identifying an expansion followed by an acronym inside of the parentheses, they will also identify other patterns that do not include parentheses, such as those based on hyphens ("-"). These patterns are using the following construction EXPANSION - ACRONYM or ACRONYM - EXPANSION. Moreover, it will add support for the Bulgarian language to this system. This improvement will include expanding the system to support the Cyrillic alphabet as well as adding a Bulgarian data set that will be used to train the algorithm used by the system. These techniques will be language-independent and should work with both English and Bulgarian. The functionality of the current system will be used as a baseline to evaluate the performance of the features mentioned above. To accomplish this, the following research question is formulated:

*What is the impact on the recall, precision, and F1-measure of the in-expansion benchmarks including techniques for AI specifically designed for the Bulgarian language when using the newly introduced Bulgarian data set?*

In order to support the main research question, the following subquestions are formulated:

- *How well is the system performing in terms of speed, recall, precision, and F-1 measure when it uses the newly introduce no-parentheses algorithm compared to the existing algorithms in the system?*
- *To what extent the new no-parentheses algorithm influences the speed of the system when the Bulgarian language is used for acronym expansion compared to when it operates in English?*

The remainder of this thesis is structured as follows. Section **??** focuses on state-of-art research that is related to the acronym expansion. Section 3 describes the methodology that will be used in this thesis, while Section 4 shows the experiment behind the study. Section 5 presents the results from it, and discusses possible

limitations. Section 6 introduces a discussion of the findings and in Section 7, the author concludes the research.

## 2 RELATED WORK

This section focuses on the existing system AcX in Section 2.1, existing research in the field of acronym identification in Section 2.2, and acronym disambiguation in Section 2.3. Moreover, it introduces previous work on the Bulgarian language in the world of Natural language processing (NLP) in Section 2.4.

### 2.1 Acronym eXpansion System

The system that this thesis contributes to is called Acronym eXpansion System, in short AcX [10]. This is an open-source system designed to expand acronyms available in a text. Its goal is to enhance acronym expansion techniques and facilitate experimentation with diverse methods for advancement.

To find acronyms and expansion in the text, the system has two main processes: Acronym Identification and Acronym Disambiguation. In the AI process, rule-based algorithms, such as MADDOG from Veyseh et al. [22] and Schwartz and Hearst [19], are employed to detect acronyms and their corresponding expansions within a document. Training documents are utilized to establish an extensive database of in-expansion pairs.

During the AD process, the system focuses on the expansions that are not found in the given document. This is achieved by encoding a set of training documents into vector representations using techniques like Doc2Vec, BERT [8], or SciBERT [3]. These representations are then combined with the in-expansion results. A predictor mechanism calculates document similarity to correctly match out-expansion acronyms with their appropriate expansions.

### 2.2 Acronym identification

Acronym identification is considered when the expansion of the acronym is available in the text. A considerable amount of literature has been published on this topic showing the two main approaches using rule-based and machine learning (ML) algorithms [13, 17, 19, 21]. Most of the studies are using the notion of precision and recall to evaluate their proposed algorithms [13, 17, 19, 21]. Precision is the number of correctly retrieved acronym-expansion pairs divided by the total number of retrieved acronym-expansion pairs. Whereas the recall is the number of correctly retrieved acronym-expansion pairs divided by the total number of acronym-expansion pairs in the data set [17].

Pustejovsky et al. [17] proposed two solutions to the AI problem using regular expressions(90% precision and 63% recall) and syntactic constraints (99% precision and 61.9% recall). The approach of syntactic constraints was used on a pre-processed data set annotated with additional syntactic information. In this way, the authors were tagging strings and phrases, which allowed them to highly constrain the context where they would have to look for acronym expansion. Moreover, the paper introduced the Medstract Gold Standard Evaluation corpus used for the evaluation of acronym expanders in the medical domain. However, the algorithm focuses on identifying acronyms by looking only at the left-hand context, such as ACRONYM (EXPANSION). The study did not consider right-hand context, where the acronym is defined first and the expansion comes later. For example, PIN - Personal Identification Number is the right-hand context, while Personal Identification Number - PIN is the left-hand context. This had a critical impact on the recall level resulting in only about two-thirds.

Furthermore, Schwartz et al. [19], introduced an algorithm that can detect acronyms using both left-hand and right-hand context, if parentheses are used to define those. The algorithm creates a set of pairs of acronyms and expansions and uses iterations to check which is the most accurate pair. The proposed solution achieved 95% precision and 82% recall on the MEDLINE data set - 1000 MEDLINE abstracts randomly selected by using the search query for the term "yeast".

Sohn et al. [21] proposed an automated evaluation of the accuracy of the acronym that selects the most probable pair of acronyms and expansion. The authors have defined seventeen different strategies of how an acronym can be matched with expansion. The proposed system achieved 97% precision and 85% recall on the Medstract Gold corpus.

Kuo et al. [13] used machine learning to achieve 95.86% precision at 84.64% recall on the AB3P corpus. The authors of the paper use the feature extraction technique to construct a feature vector consisting of 4 distinct feature sets. These features are used to train four different ML algorithms - Support Vector Machine (SVM), Naive Bayes, Logistic Regression, and Monte-Carlo Sampling Logistic Regression. Moreover, the authors introduced the BIOADI corpus containing 1200 abstracts from the medical domain, on which their algorithm achieved 93.52% precision at 79.95% recall.

Veyseh et al. [22] proposed an acronym identification model inspired by Schwartz [19]. The study expands the algorithm by adding more rules to it. The Acronym Detector rule considers all of the words with at least 60% of their letters to be upper-cased and the number of characters between 2 and 10 to be an acronym. The authors also added a rule where they consider only capitalize words to be part of the algorithm that looks for algorithms. Moreover, a rule which accounts for a match between the characters in the acronym and the words in the expansion was introduced. The algorithm achieved 89.98% precision at 87.56% recall for identifying acronyms and 96.45% precision at 79.53% recall for identifying acronym expansions. These results were based on the SciAI benchmark data set [16].

### 2.3 Acronym Disambiguation

Acronym Disambiguation is needed when the acronym expansion is not available in the text and the acronym has more than one possible expansion. In this case, the system will need to determine the right expansion according to the context.

Charbonnier et al. [5] proposed an unsupervised way to disambiguate acronyms. The research uses a word embedding approach to the AD problem. A filtered version of the full text of the documents is used by the authors in their database. The final version of the text excludes data from the NLTK stopword list, and words that are less than two characters or occur less than 5 times in the corpus. The algorithm achieves 84% accuracy on small contexts and 86% accuracy on larger contexts from the NOA Corpus.

MadDog [22] is a system that requires two resources, a dictionary that holds an acronym and expansion and a model that can find

the right expansion based on the context of the data. They have obtained the dictionary by exploiting their AI rule-based model described in Section 2.2 on different domains (Wikipedia, Arxiv papers, Reddit submission, Medline abstracts, and PMC OA subset). The authors created a dictionary that contains 426 389 unique acronyms and 3 781 739 unique expansions. A supervised model was trained on the Massive Acronym Disambiguation (MAD) data set to predict the correct expanded form of the acronym. The system achieved 92.27% precision at 85.01% recall on the SciAD data set.

## 2.4 NLP methods for the Bulgarian language

This section describes different methods for processing the Bulgarian language.

Simov et al. [20] introduced the BulTreeBank Project, in which the main goal was to prepare the Bulgarian language for automatic processing by creating a syntactically annotated data set using morphosyntactic tagging, named entity recognition, part-of-speech disambiguator, and partial parsing. To create the national corpus, the authors created three subsets of data - a core set of sentences, a treebank, and a text corpus. The core set of sentences contains more than 2500 sentences extracted from grammar books, whereas the treebank refers to syntactically annotated sentences with a total word count of around one million. The text corpus gathers a wide document collection from multiple domains. It is multi-layered linguistic annotated and its size is over a hundred million words.

Kapukaranov et al. [11] performed sentiment analysis for movie reviews in Bulgarian. They used textual features such as words, emoticons, n-grams, and the lexicon of movie reviews for sentiment analysis. In addition to those the study also used contextual features such as movie length, country, genres, actors, director, average user rating, IMBD score, and Cinexio score. The paper focuses on adding a new data set for movie reviews in Bulgarian containing more than 10000 reviews. The study used three different techniques for processing the annotated data set:

- Classification - SVM with linear kernel
- Regression - same SVM tool, features, and parameters as the classification technique but with predicted numerical value - support vector regression
- Ordinal regression - tries to fit the data into regions but at the same time predicts the values and position in the space.

Out of the above-mentioned algorithms, the regression in combination with contextual and textual features performed most optimally.

Cherecharov et al. [6] proposed a web-based system for teaching the Bulgarian language. The system introduced an NLP module for the Bulgarian language that can perform a variety of tasks. These include automatic morphological analysis and synthesis, verification of syntactic agreement, automatic placing stress, and automatic processing of complex verb forms. This allows users to search for all forms of a word within the text. Moreover, the user can replace the word with different forms or with completely different words if they are within the same part of speech.

# 3 METHODOLOGY

In this section, the author will discuss the exploratory data analysis, the annotation system that was built for the project, as well as the Bulgarian data set that was created for the project. Moreover, this section focuses on the algorithm that is used in the research.

## 3.1 Exploratory Data Analysis

At the moment of writing this paper, there is no available Bulgarian data set that is multi-domain and could be used for such an experiment. Therefore, the author of the paper has created one, that has been built out of 102 documents. The dataset is constructed using the full documents, excluding, tags such as images, links, tables, and figures. The documents were collected from Wikipedia - Bulgaria[1]. The extraction of the articles was completely random and the author did not consider any factors while picking out the articles. The randomness of the data set generation process purpose is to ensure both the diversity and domain neutrality of the data set. The author manually picked, saved, and extracted the documents using the WikiExtractor [2]. The WikiExtractor uses the WikiDump to extract the documents and saves them in a *.txt* format.

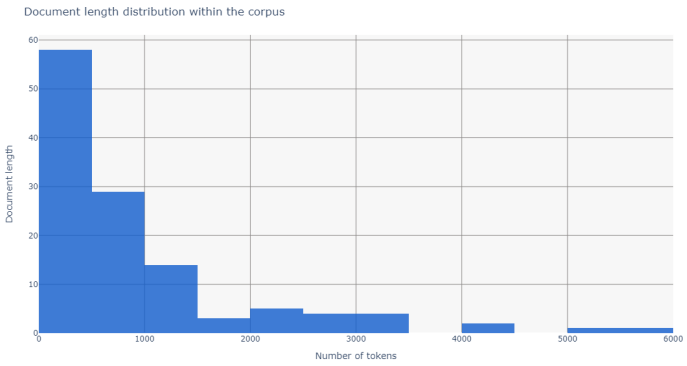


**Figure 1: Distribution of the length of the articles used in developing the Bulgarian data set**

Figure 1 shows a histogram of the distribution of the length of the data within the corpus. As could be seen in the graph, most of the articles contain less than 1500 tokens. Nevertheless, some data points contain a significantly higher amount of tokens which can be categorized as outliers. The largest document in the dataset has between 5000 and 6000 tokens. Commonly, one would think that a larger document would contain more acronyms, but this is not always the case. Moreover, the long documents could be more challenging for the annotators since the task takes way longer. This could lead to mistakes, and it might have a negative impact on the quality of the data set.

In order to prevent potential challenges encountered during the annotation process, the researcher and his colleagues built an Acronym Expander Annotation System. The documents mentioned above were later annotated by the author and external readers using the newly created system.

[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

## 3.2 Acronym Expander Annotation System

To be able to use the gathered data for this research, it needs to be pre-processed. For this purpose, 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 is accessible by annotators at https://acx-system.uc.r.appspot.com/.

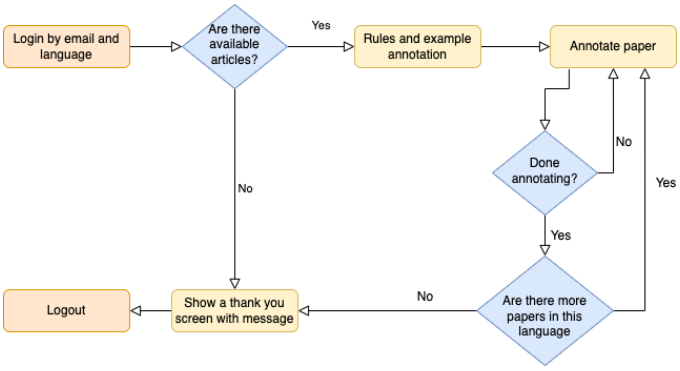


**Figure 2: Annotation system workflow**

Figure 2 shows the workflow of the system. The system allows external readers to annotate the collected data. When the user goes to the homepage, the system asks him/her to log in using email and language. Then it checks if there are any available documents for annotation, if so, it provides the user with an example of annotating a document. If there are no documents available, it logs out the user. If a document is present, the user proceeds with the annotation process. This implies that the system provides the user with an extracted text, prompting them to accurately link each acronym with its corresponding expansion, if available. 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 each identified acronym. The system gives specific instructions 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. Also, given the fact that each article should be annotated by two unique annotators, the system prioritizes unannotated articles or ones that received annotation only once. Furthermore, the system uses a smart article lock process, which assigns a document to an 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.

[3] https://python.org
[4] https://flask.palletsprojects.com/

The saved acronym expansion collection will be then reviewed, merged, and converted to a data set used to test the system.

## 3.3 Data sets in AcX

There are in total 7 data sets that are used to evaluate the performance of the AcX system. Out of these 6 are in English, and the last one is the newly introduced data set by the author, in Bulgarian (Section 3.3.7). The first 5 data sets below are used exclusively for in-expansion experiments within the AcX system, meaning their annotations only include acronym-expansion pairs with expansions present in the text. The English annotated data set (Section 3.3.6) includes acronym-expansion pairs where the expansion is not explicitly mentioned in the text documents.

*3.3.1* ***Ab3P.*** It is a medical data set, that contains 1,250 randomly selected MEDLINE articles. Initially annotated by Sohn et al.[21], the system integrates the revised version by Doğan et al [9], including 1,233 acronym-expansion pairs.

*3.3.2* ***BIOADI.*** It is a medical data set, that includes 1,201 abstracts from the BioCreative II gene, including 1,720 acronym-expansion pairs. Originally annotated by [13], the system uses the revised version by Doğan et al [9].

*3.3.3* ***Medstract.*** This is a medical data set that includes 199 MEDLINE abstracts randomly obtained from a "gene" query. Annotations were initially made by multiple researchers [1, 17, 19, 23], and the system uses the final version by Doğan et al.[9] that contains 159 acronym-expansion pairs.

*3.3.4* ***SDU-AAAI SciAI.*** The dataset comprises 6,786 English arXiv papers, with 9,775 annotated acronym-expansion pairs. It includes 17,506 sentences, with 1% lacking acronyms and 24% lacking expansions. The system utilizes the SDU@AAAI competition [4] version proposed by Veyseh et al. [22].

*3.3.5* ***Schwartz and Hearst.*** This is a medical data set that includes 1,000 randomly selected MEDLINE abstracts on query of "yeast." Originally annotated by Schwartz and Hearst [19] and revised by Doğan et al.[9], it contains 979 acronym expansion pairs.

*3.3.6* ***Annotated English Wikipedia Data set.*** Pereira et al. [10] developed a dataset of 163 randomly selected English Wikipedia documents from the Computing category. It contains 1,139 acronym-expansion pairs. Two student volunteers annotated each document, with conflicts resolved by the authors. The Inter-Annotator Agreement (IAA)[12] using Krippendorff's alpha [15] is 0.68 for in-expansion pairs and 0.33 for out-expansion pairs, indicating more disagreement on out-expansions due to the need for external text sources.

*3.3.7* ***Annotated Bulgarian Wikipedia Data set Bulgarian.*** The final annotated set consists of 102 documents that were annotated twice by 9 different annotators. The author reviewed and validated all of the annotations, and later created the final data set. This process included cleaning the annotations by adjusting a code written by other participants in the project. The code was adjusted to handle the Cyrillic alphabet, as well as, to prepare the data set according to the AcX system standard. The data set consists of in-expansion and out-expansion pairs found in the selected

documents. These are 1465 pairs in total, separated in Bulgarian, English, and other languages.

Figure 3 shows that out of these, 898 pairs are in Bulgarian (61.3%) - 163 in-expansions (18.2%) and 735 out-expansions(81.8%), 533 are in English (36.4%) - 78 in-expansions (14.6%) and 455 out-expansions (85.4%), and 34 pairs are in other languages (2.3%) - 13 in-expansions (38.2%) and 21 out-expansions (61.8%).

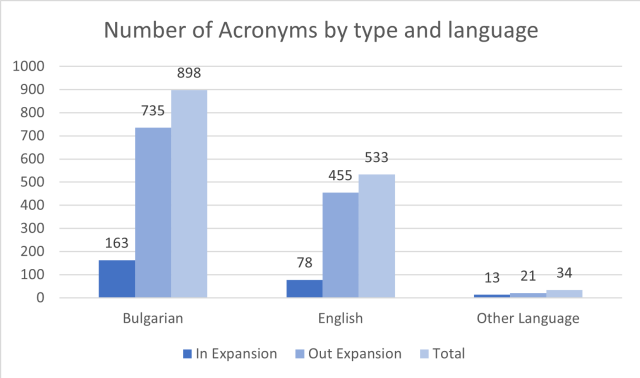


**Figure 3: Number of acronyms by type and language**

However, not all of these pairs are unique. Figure 4 shows that in the Bulgarian pairs, 195 (21.7%) are unique, while in the English pairs, the uniqueness is higher with 204 pairs (38.3%). In the case of the other language, there is a much higher percentage of unique pairs - 21 (61.7%).

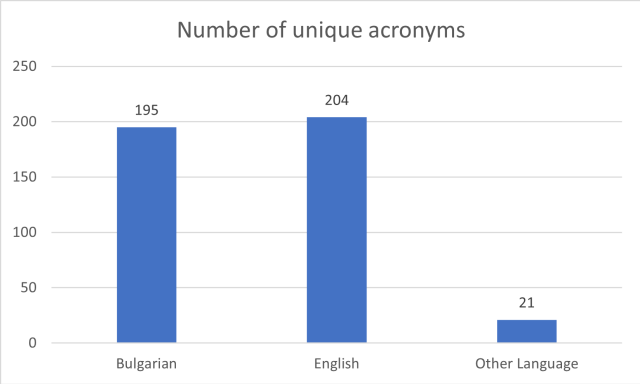


**Figure 4: Number of unique acronyms in the dataset**

Krippendorff's alpha was used to define the Inter Annotator Agreement (IAA) with MISA in the dataset. The value of the alpha for the in-expansion pairs is 0.875082, and for the out-expansion pairs is 0.676086. The measurement was conducted to assert annotation and its given instructions. Since Krippendorff's alpha is high in both of the measures, it could be concluded that the instruction given to the annotators were clear and well understood.

## 3.4 Algorithm

In this section, the author introduces the algorithm that is used to perform the evaluation of the system in the experiment.

The algorithm is separated into 4 parts:

- Tokenization of the text and extraction of the potential acronyms in it
- Preparation of tokens for expansion check
- Expansion check
- Return the global acronym map

Lines 1 to 12 describe the first part of Algorithm 1. In these steps, the text is processed and transformed into tokens. Moreover, a potential global acronym map is created based on the rules mention from line 4 to line 7. Those rules are discussed in Section 3.4.1 *Rule set for Acronym Extraction.*

Line 12 to line 26, presents the process of determination of the eligibility of a token to be a potential expansion for a given acronym. Lines 21 through 25 define the frame of tokens where a possible expansion could be located. With the use of the frame instead of processing the entire sentence the algorithm efficiency will be significantly enhanced.

Line 27 references the expansion check process (Algorithm 2). In this section, a potential expansion is determined as a viable candidate for the specific acronym. Each of the rules in this part is exclusive, therefore if the expansion fails on one of them, it is disregarded. These rules are discussed in more detail in Section 3.4.2 *Rule set for Expansion Check.*

Line 30, returns the finalized global acronym map. This process checks if an expansion has been identified for a given acronym. Assuming the expansion does not match any acronym will result in its removal from the map.

The algorithm is used in the experiment as a stand-alone algorithm, as well as in addition to the existing rule-based algorithms [16, 19].

### 3.4.1 *Rule set for Acronym Extraction.*

- *Line 4, Algorithm 1:* ***Short words*** Ignore all words that are with a length of fewer than 2 characters and that are not uppercase. These should be ignored since they cannot represent a valid acronym.
- *Line 5, Algorithm 1:* ***70% capital letters/digits*** An acronym should consist of at least 70% capital letters or digits, in any other case the word is ignored.
- *Line 6, Algorithm 1:* ***Long words*** Ignore all words that are more than 10 characters.
- *Line 7, Algorithm 1:* ***Initial Capital Letter*** Ignore all words that are not starting with a capital letter or with a digit.

### 3.4.2 *Rule set for Expansion Check.*

- *Step 1, Algorithm 2:* ***First and Last check*** Confirm that the first letter of the first and the last word of the expansion match the first and the last letter from the acronym.
- *Step 2, Algorithm 2:* ***Letter Matching*** Check that the first letters of the acronym match the first letter of the words of the expansion
- *Step 3, Algorithm 2:* ***Stop words*** Check if there are any stop words in the expansion and keep them in the *stopwords* variable
- *Step 4, Algorithm 2:* ***Acronym in Expansion check*** Check if the acronym is not in the expansion already

**Algorithm 1:** Pseudocode for acronym expansion without the need of parenthesis

**Input** : text
**Output** : acronym map

**STEP 1:** Extract all acronyms from the text, store them in *acronym_global_map*
**for** *token in (text)* **do**
    #skip rule to filter out eligible for acronym tokens
    **if** *(token.length < 2 and not token.isupper()) or*
    *(token.uppers().length < token.length *0.7) or*
    *(token.length >= 10) or*
    *(not (token[0].isdigit() or token[0].isupper()))* **then**
        continue
    #The filtered tokens are added to the global acronym map
**STEP 2:** *text_split = text.split("\n")* #Separate text on paragraphs
**STEP 3:** Separate the paragraphs on sentences and tokenize the sentences
**for** *paragraph in (text_split)* **do**
    **for** *sentence in sent_tokenize(paragraphs)* **do**
        *tokenized_text = tokenizer(sentence)*
        *acronym_map = create_acronym_map(tokenized_text)*
        **for** *acronym_index, acronym in acronym_map.items()* **do**
            **if** *acronym in acronym_global_map* **then**
                continue
            *acronym_length*= Length of the acronym expansion in words
            #Get the number of capital letters and the number of small letters that are not concatenated
            *first_capital_letter = acronym*.words()[0][0]
            *last_capital_letter = acronym*.words()[*acronym_length*-1][0]
            #Those are supposed to match the first and the last word of the expansion
            *distance* = maximum distance where the expansion could be located #Maddog
            *left_index* = index of the first word of the expansion
            *right_index* = index of the last word of the expansion
            #*long_form* is the part of the sentence that matches the *left_index* and *right_index*
            **if** *expansion_check(acronym, long_form)* #Algorithm 2 *is not None* **then**
                *acronym_global_map[acronym] = long_form*
**STEP 4:** Return the global acronym map

**Algorithm 2:** Pseudocode for expansion check process

**Input** : *acronym. long_form*
**Output** : None|expansion

**STEP 1: if** *not(long_form[0][0] === acronym[0] and long_form[last_word][0] === acronym[last])* **then**
    return None
**STEP 2: for** *letter in acronym.words()* **do**
    **if** *letter !== long_form[index][0]* **then**
        return None
**STEP 3: if** *acronym.words() in stopwords* **then**
    *stopwords++*
**STEP 4: if** *acronym in long_form* **then**
    return None
**STEP 5: if** *acronym.length() + stopwords.length() > long_form.words()* **then**
    return None
**STEP 6:** Return expansion

- *Step 5, Algorithm 2:* ***Length check*** Check if the length of the Acronym letters + the number of stop words is less than the first letters of the acronym

## 4 EXPERIMENT

This section introduces the experimental setup used to evaluate the study and the experimental metrics used in it.

### 4.1 Experimental setup

In order to perform the experiment, the author has used the existing AcX system. The system has been deployed to a CUDA GPU [5] server provided by UvA. The system has a minimum requirement of Python 3.7, which needs to be installed on the server, and it is compatible with any Unix-based Operating System (OS). The AcX team has provided the required steps that need to be performed in order to get the system up and running [6].

The thesis focus on in-expansion methods and this is why the author has used the currently available 4 algorithms for this as a baseline. These include the rule-based - Schwartz & Hearst (SH) and MadDog, and the machine learning based - SciBERT and SciDr-in. The algorithm introduced in section 3.4 is introduced in the system and it is loaded as a separate acronym expansion option, using the *python3 path/to/benchmark* **--in_expander=*schwartz_hearst_extended***. It could be used in 4 different modes:

- As a standalone algorithm, ***schwartz_hearst_extended*** (Iliya)
- Extension to SH, ***schwartz_hearst_extended_sh_iliya*** (SH + Iliya)
- Extension to MadDog, ***schwartz_hearst_extended_maddog_iliya*** (MadDog + Iliya)

[5] CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs https://developer.nvidia.com/cuda-zone
[6] https://web.tecnico.ulisboa.pt/~ist164790/acx/

- Extension to both SH and MadDog, ***schwartz_hearst_extended_sh_maddog_iliya*** (SH + MadDog + Iliya)

All of the rule-based algorithms are performed on a regular CPU on the GPU server, while the ML algorithm is run on the Cuda GPU environment. In order to get the timing of each of the algorithms, the author took 3 consecutive runs and averaged the results. All of the algorithms are run against the already available data sets in the system, as well as the newly introduced Bulgarian data set discussed in section 3.3.7.

Moreover, the experiment focuses only on acronym identification, and therefore the out expansion is not taken into consideration for the results of this study.

## 4.2 Experimental metrics

To evaluate the results of this study, the author will use the standard for the field 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 algorithm to process a single article

These metrics have been used in numerous different types of research from the acronym expansion domain [10, 19, 22]. Since the research is based on acronym identification the most important measure that the author observes is recall. The newly introduced algorithm should identify more acronyms in the text due to the less restricted method of identifying the acronym candidates.

# 5 RESULTS

This section discusses the in-expansion results of the experiment. This refers to the comparison of the results between the existing algorithms in the system and the recently introduced algorithm. The results of the study are collected manually by the author by running the benchmark scripts for the given data set while using the different in-expander options in the AcX system. The algorithms in the system are labeled with numbers from 1 to 7, where algorithms 1 to algorithm 6 are rule-based, and algorithm 7 is ML.

## 5.1 Medical data sets

Table 1 displays the average values of the experimental metrics for the four discussed medical data sets. The results indicate that, on average, Schwartz-Hearst (1) demonstrates superior performance compared to other algorithms in *Precision*, exhibiting an advantage of up to 0.002% in Acronyms in text and 0.024% in Pairs in text. The combination of SH + MADDOG and Iliya (6) surpasses other algorithms with a *Recall* value of 0.014% and an *F1 score* of 0.02% in Acronyms in text. In the Pairs in text category, it achieves higher performance with values of 0.015% and 0.008%, respectively. Table 5 in Appendix A shows a more detailed analysis of the benchmarks conducted on all medical data sets. This table provides comprehensive insights into the performance of all algorithms within the system across different data sets. It is observed that MADDOG (2) and Schwartz-Hearst (1) yield the highest *Precision* scores for both Acronyms in text and Pairs in text, while, the combination of SH + MADDOG and Iliya (6) outperforms other algorithms in terms of *Recall* and *F1 Measure*.

| Algorithm | Average medical data sets | | | | | | |
|---|---|---|---|---|---|---|---|
| | Acronyms in text | | | Pairs in text | | | Exec Time |
| | P | R | F1 | P | R | F1 | |
| Schwartz-Hearst (1) | **0,993** | 0,819 | 0,897 | **0,963** | 0,795 | 0,871 | **0,0003** |
| MADDOG (2) | 0,984 | 0,586 | 0,734 | 0,923 | 0,550 | 0,688 | 0,033 |
| Iliya (3) | 0,963 | 0,330 | 0,486 | 0,877 | 0,305 | 0,447 | 0,004 |
| SH + Iliya (4) | 0,955 | 0,842 | 0,895 | 0,918 | 0,810 | 0,860 | 0,007 |
| MADDOG + Iliya (5) | 0,965 | 0,618 | 0,752 | 0,900 | 0,577 | 0,702 | 0,040 |
| SH + MADDOG + Iliya (6) | 0,969 | **0,849** | **0,905** | 0,937 | **0,821** | **0,875** | 0,039 |
| SciBert (7) | 0,856 | 0,696 | 0,767 | 0,683 | 0,556 | 0,613 | 0,811 |

**Table 1: Average performance of medical data sets in the AcX system**

In terms of *execution time*, the Schwartz-Hearst (1) algorithm was the most efficient, while the standalone algorithm of Iliya (3) came in second. The SH + MADDOG + Iliya (6) algorithm came in fifth with 0,039s per article.

## 5.2 Non-medical data sets

Table 2 presents the benchmark results for the non-medical data sets. Regarding *Precision* in Acronyms and Pairs in text, MADDOG (2) achieved the highest scores with 0.026% and 0.04% for SDU, and 0.013% and 0.006% for the Wikipedia data set, respectively. However, when examining the *Recall* and *F1-measure*, it becomes clear that an extended algorithm achieves the highest values.

For SDU, the combination of SH + MADDOG + Iliya (6) performed best in terms of Acronyms in text, while the combination of MADDOG + Iliya (5) was the most effective for Pairs in text. In terms of *F1 Measure* for SDU, Scibert yielded the best results for Acronyms, and MADDOG + Iliya (5) outperformed others for Pairs in text. In the case of the Wikipedia data set, SH + Iliya achieved the highest *Recall* and *F1-measure* scores for both Acronyms and Pairs in text, while MADDOG + Iliya (5) performed best in *F1-measure* for Acronyms.

Regarding *execution times*, a similar trend to the medical data sets benchmark was observed, with Schwartz-Hearst being the fastest algorithm, while the remaining algorithms exhibited comparable speeds. SciBert (7) was the slowest-performing algorithm.

## 5.3 Newly introduced Bulgarian Wikipedia data set

The benchmark results for the newly introduced Bulgarian data set are presented in Table 3. The table illustrates the performance of various algorithms across different metrics. MADDOG (2) demonstrates the highest *Precision* in both data set metrics, outperforming other algorithms by 0.043% and 0.026%. In terms of *Recall* for acronyms in the text, the combination of SH + Iliya (4) achieves

| Algorithm | SDU | | | | | | | Wikipedia EN | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acronyms in text | | | Pairs in text | | | Exec Time | Acronyms in text | | | Pairs in text | | | Exec Time |
| | P | R | F1 | P | R | F1 | | P | R | F1 | P | R | F1 | |
| Schwartz-Hearst (1) | 0,960 | 0,824 | 0,887 | 0,929 | 0,796 | 0,857 | **0,0001** | 0,901 | 0,602 | 0,722 | 0,858 | 0,573 | 0,687 | **0,0003** |
| MADDOG (2) | **0,986** | 0,867 | 0,923 | **0,969** | 0,852 | 0,907 | 0,008 | **0,949** | 0,591 | 0,728 | **0,911** | 0,567 | 0,699 | 0,140 |
| Iliya (3) | 0,937 | 0,886 | 0,911 | 0,917 | 0,867 | 0,891 | 0,001 | 0,936 | 0,544 | 0,688 | 0,905 | 0,525 | 0,664 | 0,013 |
| SH + Iliya (4) | 0,918 | 0,932 | 0,925 | 0,885 | 0,899 | 0,892 | 0,001 | 0,839 | **0,691** | 0,766 | 0,807 | **0,649** | **0,719** | 0,013 |
| MADDOG + Iliya (5) | 0,937 | 0,929 | 0,933 | 0,916 | **0,908** | **0,912** | 0,011 | 0,923 | 0,660 | **0,769** | 0,878 | 0,628 | 0,732 | 0,155 |
| SH + MADDOG + Iliya (6) | 0,926 | **0,936** | 0,931 | 0,894 | 0,903 | 0,898 | 0,011 | 0,872 | 0,681 | 0,764 | 0,818 | 0,639 | 0,717 | 0,151 |
| SciBert (7) | 0,966 | 0,925 | **0,945** | 0,914 | 0,875 | 0,894 | 0,404 | 0,765 | 0,591 | 0,667 | 0,693 | 0,536 | 0,604 | 0,968 |

**Table 2: Comparison between SDU AAAI and Wikipedia EN data sets**

the best performance, with a slight 0.008% difference from SH + MADDOG + Iliya (6). The MADDOG + Iliya (5) algorithm yields the highest *F1 measure*, surpassing Iliya (3) by 0.007%. Similarly, MADDOG + Iliya (5) achieves the best recall for Pairs in text, while the standalone Iliya (3) algorithm outperforms others in the *F1 measure*.

Consistent with previous benchmarks, Schwartz-Hearst (1) achieved the best execution time, followed by Iliya (3).

| Algorithm | Wikipedia BG | | | | | | |
|---|---|---|---|---|---|---|---|
| | Acronyms in text | | | Pairs in text | | | Exec Time |
| | P | R | F1 | P | R | F1 | |
| Schwartz-Hearst (1) | 0,824 | 0,636 | 0,718 | 0,758 | 0,585 | 0,660 | **0,0004** |
| MADDOG (2) | **0,906** | 0,653 | 0,759 | **0,859** | 0,619 | 0,719 | 0,161 |
| Iliya (3) | 0,863 | 0,746 | 0,800 | 0,833 | 0,720 | **0,773** | 0,016 |
| SH + Iliya (4) | 0,782 | **0,788** | 0,785 | 0,731 | 0,737 | 0,734 | 0,019 |
| MADDOG + Iliya (5) | 0,836 | 0,780 | **0,807** | 0,800 | **0,746** | 0,772 | 0,184 |
| SH + MADDOG + Iliya (6) | 0,760 | 0,780 | 0,770 | 0,711 | 0,729 | 0,720 | 0,182 |
| SciBert (7) | 0,462 | 0,364 | 0,408 | 0,290 | 0,229 | 0,257 | 1,560 |

**Table 3: Comparison of the efficiency of the existing algorithms on the newly introduced Bulgarian data set**

## 5.4 Overall performance per data set

| Data set | Acronyms in text | | | Pairs in text | | | Exec Time |
|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | |
| Ab3P | 2 | 6 | 6 | 2 | 6 | 6 | 1 |
| BIOADI | 1 | 6 | 6 | 1 | 6 | 1 | 1 |
| Medstract | 1,2,3,5,6 | 6 | 6 | 1,6 | 6 | 6 | 1 |
| Average medical | 1 | 6 | 6 | 1 | 6 | 6 | 1 |
| SH | 1 | 4 | 4,6 | 1 | 4,6 | 6 | 1 |
| SDU | 2 | 6 | 5 | 2 | 5 | 5 | 1 |
| Wikipedia EN | 2 | 6 | 5 | 2 | 4 | 5 | 1 |
| Wikipedia BG | 2 | 4 | 5 | 2 | 5 | 3 | 1 |

**Table 4: Overall performance of the AcX algorithms per data set**

Table 4 provides a detailed overview of the algorithm performance in the AcX system across all of the data sets. It highlights the best-performing algorithms in terms of "Acronyms and Pairs in-text" as well as their *execution time*. The table suggests that MADDOG(2) and Schwartz-Hearst (1) are the best in terms of *Precision*, ranking first in 9 and 8 benchmarks, respectively. In the domain of *Recall*, SH + MADDOG + Iliya (6) performs as the best algorithm, achieving the first position in 10 benchmarks, followed by SH + Iliya (4) with 4 first positions. When considering the *F1 measure*, SH + MADDOG + Iliya (6) again takes first position, being the best in 9 benchmarks, while MADDOG + Iliya (5) ranks second with 5 benchmarks. Last but not least, in terms of *execution time*, Schwartz-Hearst (1) dominates as the fastest algorithm across all benchmarks.

## 5.5 Limitations of the study

This subsection reviews the limitations that were encountered in the study and suggests what could be further researched in future works in this study.

*5.5.1* ***Lack of data sets in the system***. Currently, the system offers data sets exclusively in English and Bulgarian, which constrains the scope of the benchmarks in the system. Moreover, it hinders a comprehensive evaluation of the algorithms in the system across diverse languages and lexical rules. For future research, it is vital to incorporate additional languages into the system. This expansion would enable comparative analyses of the performance of the different rule sets across different new languages and it would facilitate the identification of correlations in the usage of different rule sets across diverse alphabets.

*5.5.2* ***The size of the Bulgarian data set***. The Bulgarian data set compiled for this study is limited due to the lack of time and resources. Moreover, this data set was created by a single person and the outcome is that the corpus is relatively smaller compared to the rest of the available data sets in the system. Furthermore, the quality of the data set could have been impacted because volunteers annotated the data.

# 6 DISCUSSION

The goal of this paper was to evaluate the impact of introducing new AI algorithms in the AcX system. These were designed mainly

for detecting acronyms that have no brackets ( ) or any other lexical marks. The study investigates the impact of the new algorithm on the recall, precision, F-1 measure, and execution time for both English and Bulgarian data sets. In this discussion, the author addresses the main research question and its sub-questions in order to provide an adequate understanding of the findings discussed in the study.

### 6.1 Impact of Techniques for the Bulgarian Language on Recall, Precision, F1-measure and execution time

The results obtained from the benchmarks using the Bulgarian data set shows the performance of the system when incorporating techniques created for the Bulgarian language. By comparing the recall, precision, and F1-measure of the in-expansion benchmarks, the author can evaluate the impact of the newly introduced algorithm with the rest in the system. As one can see the MADDOG(2) algorithm performs best in precision, however, the newly introduced algorithm has a positive impact on the recall of the benchmark since it should detect more acronyms due to the less strict rules for the AI. This could be observed in the results of the study in Table 3. In terms of execution time, Schwartz and Hearst (1) performed best in the benchmark of the Bulgarian data set. This is due to the fact, that the SH (1) algorithm is simplistic and does not take into consideration as many rules as the rest of the algorithms.

These findings indicate that the newly introduced Bulgarian data set, although limited in size, provides valuable insights into the performance of the system in the Bulgarian language context. Despite the limitations, the data set enabled the evaluation of various algorithms and their effectiveness in acronym expansion tasks in Bulgarian.

### 6.2 Performance of the No-Parentheses Algorithm

To support the main research question, the author assessed the performance of the system when using the new no-parentheses algorithm compared to existing algorithms in the system. Specifically, the study examined the system's speed, recall, precision, F1 measure, and execution time per model.

The results revealed that the no-parentheses algorithm showed promising performance in terms of recall and F1 measure. It demonstrated superior recall metrics, particularly in identifying acronyms and their expansions within all the data sets with the exclusion of BIOADI where Schwartz and Hearst (1) had a better F1 measure. Moreover, the algorithm showed significantly better results without compromising as much on the execution time, falling behind 0,008s on average behind MADDOG (2) and coming second on all data sets when used as a standalone algorithm.

### 6.3 Influence of the new algorithm on execution time on the Bulgarian data set compared to the English data sets in the system

The author further investigated the influence of the no-parentheses algorithm on the speed of the system when operating in the Bulgarian language compared to English. The analysis aimed to understand whether the new algorithm impacted the system's efficiency.

The results indicated that while the no-parentheses algorithm contributed to improved recall and F1-measure, its impact on system speed varied across the language contexts. In the Bulgarian language, the algorithm exhibited a comparable processing speed to the existing algorithms in the system. This suggests that the new algorithm did not significantly differ in execution time from the rest of the system.

### 6.4 Overall findings

The findings suggest that incorporating techniques specifically designed for the Bulgarian language, such as the no-parentheses algorithm, positively influenced the recall and F1 measure of the in-expansion benchmarks, however, it had a negative impact on the precision. This is due to the fact that the algorithm generated more false positives due to the loose rule set of AI. Based on this, one could decide what algorithm should be used based on the use case required for the given scenario. If it is more important that the system yields correct acronyms and expansions, then one should use MADDOG (2) as an in-expansion algorithm. However, if one should find it important to retrieve as much as acronyms and expansion pairs as possible, then the system should use the SH + MADDOG + Iliya (6) combination.

Despite the limitations of the Bulgarian data set, the evaluation highlighted the potential of the system in handling acronym expansion tasks in Bulgarian. The no-parentheses algorithm demonstrated favorable performance and did not significantly impact the system's speed when operating in the Bulgarian language compared to English.

However, it is essential to acknowledge the limitations of the study. The Bulgarian data set's size and annotation process might have had a certain impact on the benchmark measure. Future research should aim to address these limitations by expanding the data set and involving a larger pool of annotators to enhance data quality and reliability. In this way, the rule sets used for the study could be defined even better and this could boost the performance of the algorithm.

## 7 CONCLUSION

This paper investigated the impact of incorporating techniques specifically designed for the Bulgarian language on the performance of the AcX system in acronym expansion tasks. By utilizing the newly introduced Bulgarian data set, the system's recall, precision, F1-measure, and execution time were evaluated. The study also examined the performance of the no-parentheses algorithm and its influence on the system in both Bulgarian and English contexts.

The study pointed out that despite the limitations of the Bulgarian data set, the evaluation provided valuable insights into the system's performance in acronym expansion benchmarks, focusing on comparing precision, recall, F1-measure, and execution time

metrics. The inclusion of the no-parentheses algorithm further improved recall, without significantly affecting system speed. The system was able to effectively identify and expand acronyms within Bulgarian and English texts, resulting in improved recall and overall performance.

Future work in this area should focus on expanding the Bulgarian data set to enhance the quality of the benchmark and to be able to better assess the performance of the system. Additionally, the system should be tested with other languages to explore the different techniques across different linguistic contexts and identify correlations between acronym usage patterns in multiple alphabets and languages.

In conclusion, this study contributes to the field of acronym expansion by demonstrating the potential of incorporating language-specific techniques. The findings emphasize the need to adapt existing systems for different languages and highlight the positive impact of the no-parentheses algorithm in improving recall. By considering the specific linguistic characteristics and lexical rules of a language, acronym expansion systems can achieve higher performance.

## REFERENCES


[1] Hiroko Ao and Toshihisa Takagi. 2005. ALICE: An algorithm to extract abbreviations from MEDLINE. *Journal of the American Medical Informatics Association* 12, 5 (2005), 576–586. https://doi.org/10.1197/jamia.M1757

[2] Adrian Barnett and Zoe Doubleday. 2020. The growth of acronyms in the scientific literature. *eLife* 9 (2020), 1–10. https://doi.org/10.7554/eLife.60080

[3] 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

[4] Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, Walter Chang, and Leo Anthony Celi. 2021. Acronym identification and disambiguation shared tasks for scientific document understanding. *CEUR Workshop Proceedings* 2831 (2021). arXiv:2012.11760

[5] Jean Charbonnier and Christian Wartena. 2018. Using word embeddings for unsupervised acronym disambiguation. *COLING 2018 - 27th International Conference on Computational Linguistics, Proceedings* (2018), 2610–2619.

[6] Stoyan Cherecharov, Hristo Krushkov, and Mariana Krushkova. 2017. Nlp Module for Bulgarian Text Processing. *CBU International Conference Proceedings* 5 (2017), 1113–1117. https://doi.org/10.12955/cbup.v5.1080

[7] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (sep 1995), 273–297. https://doi.org/10.1007/BF00994018

[8] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1, Mlm (2019), 4171–4186. arXiv:1810.04805

[9] Rezarta Islamaj Doğan, Donald C. Comeau, Lana Yeganova, and W. J. Wilbur. 2014. Finding abbreviations in biomedical literature: three BioC-compatible modules and four BioC-formatted corpora. *Database* 2014 (2014), 1–7. https://doi.org/10.1093/database/bau044

[10] João L.M. João, João Casanova, Helena Galhardas, and Dennis Shasha. 2022. AcX: System, Techniques, and Experiments for Acronym Expansion. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2530–2544. https://doi.org/10.14778/3551793.3551812

[11] Borislav Kapukaranov and Preslav Nakov. 2015. Fine-grained sentiment analysis for movie reviews in Bulgarian. *International Conference Recent Advances in Natural Language Processing, RANLP* 2015-Janua (2015), 266–274.

[12] Klaus Krippendorff. 2011. ScholarlyCommons Computing Krippendorff ' s Alpha-Reliability Computing Krippendorff ' s Alpha-Reliability. (2011).

[13] Cheng Ju Kuo, Maurice H.T. Ling, Kuan Ting Lin, and Chun Nan Hsu. 2009. BIOADI: A machine learning approach to identifying abbreviations and definitions in biological literature. *BMC Bioinformatics* 10, SUPPL. 15 (2009), 1–10. https://doi.org/10.1186/1471-2105-10-S15-S7

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations ofwords and phrases and their compositionality. *Advances in Neural Information Processing Systems* (2013), 4178–4179. arXiv:1310.4546

[15] Rebecca Passonneau. 2006. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006* (2006), 831–836.

[16] Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2021. What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation. (2021), 3285–3301. https://doi.org/10.18653/v1/2020.coling-main.292 arXiv:2010.14678

[17] James Pustejovsky, José Castaño, Brent Cochran, Maciej Kotecki, and Michael Morrell. 2001. Automatic extraction of acronym-meaning Pairs from MEDLINE databases. , 371–375 pages. https://doi.org/10.3233/978-1-60750-928-8-371

[18] N. Saneesh Mohammed and K. A.Abdul Nazeer. 2013. An improved method for extracting acronym-definition pairs from biomedical Literature. *2013 International Conference on Control Communication and Computing, ICCC 2013* Iccc (2013), 194–197. https://doi.org/10.1109/ICCC.2013.6731649

[19] 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

[20] Kiril Simov, Petya Osenova, Milena Slavcheva, Sia Kolkovska, Elisaveta Balabanova, Dimitar Doikoff, Krassimira Ivanova, Alexander Simov, and Milen Kouylekov. 2002. Building a linguistically interpreted corpus of Bulgarian: The bul tree bank. *Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC 2002* (2002), 1729–1736.

[21] Sunghwan Sohn, Donald C. Comeau, Won Kim, and John W. Wilbur. 2008. Abbreviation definition identification based on automatic precision estimates. *BMC Bioinformatics* 9 (2008), 1–10. https://doi.org/10.1186/1471-2105-9-402

[22] 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

[23] Anna Yarygina and Natalia Vassilieva. 2012. High-recall extraction of acronym-definition pairs with relevance feedback. *ACM International Conference Proceeding Series* (2012), 21–28. https://doi.org/10.1145/2320765.2320781

| Algorithm | Ab3P | | | | | | | BIOADI | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acronyms in text | | | Pairs in text | | | Exec Time | Acronyms in text | | | Pairs in text | | | Exec Time |
| | P | R | F1 | P | R | F1 | | P | R | F1 | P | R | F1 | |
| Schwartz-Hearst (1) | 0,986 | 0,777 | 0,869 | 0,952 | 0,749 | 0,838 | **0,0004** | **0,991** | 0,796 | 0,883 | **0,941** | 0,756 | **0,839** | **0,0002** |
| MADDOG (2) | **0,992** | 0,643 | 0,780 | **0,954** | 0,619 | 0,750 | 0,026 | 0,980 | 0,552 | 0,706 | 0,876 | 0,493 | 0,631 | 0,040 |
| Iliya (3) | 0,932 | 0,450 | 0,607 | 0,898 | 0,433 | 0,585 | 0,004 | 0,952 | 0,301 | 0,457 | 0,886 | 0,280 | 0,425 | 0,004 |
| SH + Iliya (4) | 0,938 | 0,820 | 0,875 | 0,894 | 0,782 | 0,834 | 0,004 | 0,955 | 0,809 | 0,876 | 0,906 | 0,767 | 0,831 | 0,009 |
| MADDOG + Iliya (5) | 0,947 | 0,687 | 0,796 | 0,914 | 0,662 | 0,768 | 0,046 | 0,957 | 0,590 | 0,730 | 0,853 | 0,526 | 0,650 | 0,037 |
| SH + MADDOG + Iliya (6) | 0,944 | **0,828** | **0,882** | 0,910 | **0,798** | **0,851** | 0,047 | 0,962 | **0,820** | **0,886** | 0,909 | **0,775** | 0,837 | 0,038 |
| SciBert (7) | 0,854 | 0,730 | 0,787 | 0,694 | 0,594 | 0,640 | 0,848 | 0,860 | 0,730 | 0,789 | 0,655 | 0,556 | 0,601 | 0,754 |
| **Algorithm** | **Medstract** | | | | | | | **SH** | | | | | | |
| | Acronyms in text | | | Pairs in text | | | Exec Time | Acronyms in text | | | Pairs in text | | | Exec Time |
| | P | R | F1 | P | R | F1 | | P | R | F1 | P | R | F1 | |
| Schwartz-Hearst (1) | **1,000** | 0,891 | 0,943 | **1,000** | 0,891 | 0,943 | **0,0003** | **0,996** | 0,813 | 0,895 | **0,960** | 0,784 | **0,863** | **0,0003** |
| MADDOG (2) | **1,000** | 0,630 | 0,773 | 0,931 | 0,587 | 0,720 | 0,037 | 0,966 | 0,518 | 0,674 | 0,933 | 0,500 | 0,651 | 0,027 |
| Iliya (3) | **1,000** | 0,239 | 0,386 | 0,818 | 0,196 | 0,316 | 0,004 | 0,968 | 0,331 | 0,493 | 0,905 | 0,309 | 0,461 | 0,004 |
| SH + Iliya (4) | 0,976 | 0,891 | 0,932 | 0,976 | 0,891 | 0,932 | 0,004 | 0,952 | **0,849** | **0,897** | 0,895 | **0,799** | 0,844 | 0,009 |
| MADDOG + Iliya (5) | **1,000** | 0,630 | 0,773 | 0,931 | 0,587 | 0,720 | 0,039 | 0,957 | 0,565 | 0,710 | 0,902 | 0,532 | 0,670 | 0,039 |
| SH + MADDOG + Iliya (6) | **1,000** | **0,913** | **0,955** | **1,000** | **0,913** | **0,955** | 0,031 | 0,971 | 0,835 | **0,897** | 0,929 | **0,799** | 0,859 | 0,041 |
| SciBert (7) | 0,818 | 0,587 | 0,684 | 0,636 | 0,457 | 0,532 | 0,888 | 0,891 | 0,737 | 0,807 | 0,748 | 0,619 | 0,677 | 0,756 |

**Table 5: Detailed comparison between medical data sets available in the AcX system**

# Appendix A DETAILED COMPARISON BETWEEN MEDICAL DATA SETS AVAILABLE IN THE ACX SYSTEM