UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Towards Effective and Effortless Data Cleaning: from Automatic Approaches to User Involvement

João Pedro Lebre Magalhães Pereira

Supervisor:	**Doctor Helena Isabel de Jesus Galhardas**
Co-Supervisor:	**Doctor Dennis Elliott Shasha**

Thesis aproved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction

2023

TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Towards Effective and Effortless Data Cleaning: from Automatic Approaches to User Involvement

João Pedro Lebre Magalhães Pereira

Supervisor: Doctor Helena Isabel de Jesus Galhardas
Co-Supervisor: Doctor Dennis Elliott Shasha

Thesis aproved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction

Jury

Chairperson: Doctor Ana Maria Severino de Almeida e Paiva, Instituto Superior Técnico, Universidade de Lisboa;

Members of the Committee:

Doctor Paulo Miguel Torres Duarte Quaresma, Escola de Ciências e Tecnologia, Universidade de Évora;

Doctor Mário Alexandre Teles de Figueiredo, Instituto Superior Técnico, Universidade de Lisboa;

Doctor Laure Berti-Équille, IRD - the French National Research Institute for Sustainable Development, France;

Doctor Luís Manuel Antunes Veiga, Instituto Superior Técnico, Universidade de Lisboa;

Doctor Helena Isabel de Jesus Galhardas, Instituto Superior Técnico, Universidade de Lisboa.

Funding Institutions - Universidade de Lisboa
Fundação para a Ciência e a Tecnologia

2023

Dedicated to my son Tomás and my wife Diana.

Acknowledgments

I do not think I could ever demand all the credits for this thesis. Therefore, there are a lot of people I would like to express my gratitude to. Because of their help, I feel this is their work too.

First of all, I would like to thank my supervisor, Professor Helena Galhardas. We started working a long time a go, since I started the MSc Thesis. I thank you for your guidance in busy times, your patience when I was proceeding wrong, and by enlightening me with your experience and knowledge when I lacked fruitful ideas. I hope you are proud of this work too.

I would also like to give a special thank you to my co-supervisor, Professor Denish Shasha, whose collaboration on Acronym Expander was essential to this thesis, for all the support and the time spent helping and guiding me through discussions and for his genuine interest and for introducing me to the acronym expander topic. I also hope you are proud of this work.

I am very grateful for the opportunity to have collaborated on iterative data cleaning with Professor Antónia Lopes and Professor Manuel J. Fonseca. Their wisdom, expertise, and discussions were fundamental to build up this thesis. Certainly, without their kind participation, this work would not be of such high quality and I would be less prepared to conduct research work. I hope you too can be proud of this work and find that I've made good use of your efforts.

A special thanks goes also to the MSc students that I've worked with in the context of this thesis, specially to Cátia Ormonde for starting the work on conflict resolution of manual data repairs in data cleaning programs, and to João Casanova, who I officially had the pleasure to co-supervise, for his work on acronym and expansion extraction.

Another thanks goes to all the classmates that crossed my path. They provided me with good times that I will not forget. A special thanks to an informal friends group of successful professionals that were closely present and supportive during the entire thesis work: Fernando Santos, Eva Nave, Sara Encarnação, Vítor Vasconcelos, Flávio Pinheiro, and Diana Orghian. Another group I would like to thank and that welcomed me on some of their social events where some PhD students and professors from the graphics and agents groups at INESC-ID. Thanks to Carol Finamore for her friendship and presence.

A special thanks to João Cardoso for informing and arranging the servers matters in our group, his work was also essential.

Another set of thanks goes to the University of Amsterdam where I've been working as a Junior Lecturer since August of 2021: thanks to my colleagues, special Ashish Say for his constant motivational support towards the end of this thesis and his friendship; and my supervisor Lisa van Pappelendam for the kind welcome to the university and her support and understanding regarding the thesis work.

Much obliged to all my group of friends for their understanding when I had to work and could not hang out with them and for the support when lady luck was nowhere to be found.

To my parents, António Manuel Pereira and Maria Manuel Pereira for their support and dedication, and love, goes my sincere gratitude. It's undoubtedly due to them that I am here today, and am able to pursue my dreams. To my sister, Sofia Pereira, her fiancé Miguel Polvora and my niece Gabriela, that built a lovely family, although currently far, I thank for their love and for trying to be part of my life.

To my son and wife, Tomás Pereira and Diana Correa, to whom I dedicate this thesis, for their support and understanding during the difficult times we had to go through despite all constraints, in particular to my wife physical limitations. Thank you for bringing so much joy and strength to my life!

I also would like to acknowledge the financial support that *Universidade de Lisboa (ULisboa)* provided under a PhD scholarship awarded through the *programa de bolsas de doutoramento* 2015 and that *Fundação para a Ciência e Tecnologia (FCT)* provided under the PhD Scholarship SFRH/BD/135719/2018-EIA/115346/2009). Paramount for the development of this project. The server virtual machines used to run the experiments of this thesis were supported by: (i) BioData.pt – *Infraestrutura Portuguesa de Dados Biológicos*, project 22231/01/SAICT/2016, funded by Portugal 2020; (ii) Google Cloud scholarship for PhD students; and (iii) research grant in a form of credits to use in the research cloud platform provided by SURFsara (the Dutch national High-Performance Computing and e-Science support centre) through the University of Amsterdam.

Finally, I would like to extend my acknowledgments to everyone that in a way or another helped in the development of this thesis. Thank you all very much!

Resumo

Nos últimos anos, devido à rápida disseminação de dados e aos sensores com custo reduzido, os dados foram criados a um ritmo muito rápido. Os dados são continuamente produzidos, examinados e cada vez mais usados para tomar decisões importantes. Recomendações automáticas e tomadas de decisão com base nesses dados podem levar a enormes benefícios para a sociedade. Por outro lado, a existência de abundantes quantidades de dados aumenta a probabilidade de ocorrerem problemas de qualidade de dados com impacto negativo nas decisões baseadas em dados. Os problemas de qualidade de dados podem ser erros, valores ausentes, valores com significado duvidoso, valores duplicados e inconsistências. Um processo de limpeza de dados desempenha um papel importante na correção desses problemas. Portanto, nesta tese, abordamos dois assuntos principais: (i) expansão de acrónimos para atribuir uma expansão a acrónimos encontrados no texto de forma a melhorar a legibilidade do texto e (ii) suporte para o envolvimento do utilizador durante um processo de limpeza de dados, para reduzir o esforço de produzir efetivamente dados limpos.

A existência de acrónimos sem expansão no texto é considerado um problema de qualidade dos dados, mas que tem não tem sido examinado pela comunidade de limpeza de dados. De facto, existe a necessidade de um sistema de *software* de expansão de acrónimos disponível que possa encontrar automaticamente as expansões dos acrónimos em documentos textuais e que seja devidamente avaliado. Além disso, nos casos em que um acrónimo tem mais de uma expansão disponível, o critério geralmente aplicado para selecionar a expansão correta é a similaridade por cosseno (*Cosine similarity*) entre uma representação do documento contendo o acrónimo e uma representação de cada documento contendo uma expansão. Afirmamos que o processo para selecionar a expansão certa pode ser melhorado com outras representações de termos e documentos, assim como técnicas de aprendizagem de máquina que substituem a similaridade por cosseno.

Um processo de limpeza de dados é geralmente um processo iterativo porque pode precisar ser executado e refinado repetidamente até ser capaz de produzir a mais alta qualidade de dados possível. Além disso, devido à especificidade de alguns problemas de qualidade de dados e à limitação das regras de qualidade de dados para cobrir todos os problemas de limpeza de dados, muitas vezes um utilizador deve estar ativamente envolvido na execução de um programa de limpeza de dados reparando os dados manualmente. No entanto, não existe uma estrutura que suporte o envolvimento do utilizador no processo iterativo de limpeza de dados. Além disso, as ferramentas usadas para limpeza de dados que de alguma forma envolvem o utilizador no processo não foram avaliadas com utilizadores reais para avaliar o esforço do utilizador ao desenhar programas de limpeza de dados e ao reparar dados manualmente.

Neste trabalho contribuímos com novas abordagens que proporcionam um processo de limpeza de dados eficaz e sem esforço. Em particular, propusemos: (i) um sistema *end-to-end* para expandir acrónimos e extensível com novas abordagens de extração de pares acrónimo-expansão, e desambiguação de acrónimos, juntamente com três *benchmarks* para avaliar o desempenho de sistemas *end-to-end*, das abordagens para extrair pares acrónimo-expansão e das abordagens para de-

sambiguar acrónimos no texto; e (ii) uma *framework* melhorada para a limpeza de dados com suporte para o envolvimento do utilizador durante um processo iterativo de limpeza de dados e realizamos uma comparação experimental de ferramentas usadas para a limpeza de dados com utilizadores reais e simulados.

Finalmente, avaliamos as nossas contribuições. Especificamente, a melhor variante do nosso sistema para expandir acrónimos obteve 54,97% da medida F1 para expandir acrónimos, enquanto o único sistema presente no trabalho relacionado, *MadDog*, obteve 32,93%. Para a *framework* de limpeza de dados, com base nas respostas aos questionários do utilizador e no esforço do utilizador medido verificamos que de facto ajuda a reduzir o esforço do utilizador, obtendo pontuações mais altas do que as outras ferramentas de *software* utilizadas para a comparação (*Pentaho Data Integration* e *OpenRefine*). Por exemplo, os utilizadores especializados, no questionário de satisfação do Modelo de Aceitação de Tecnologia (*Technology Acceptance Model - TAM*) deram em média 30 em 42 pontos para facilidade de uso e utilidade. Os mesmos utilizadores, efetuaram em média 95 cliques a menos ao usar a nossa *framework* de limpeza de dados para limpar dados manualmente. Assim, concluímos que o sistema de expansão de acrónimos e a *framework* de limpeza de dados fornecem novas formas eficazes de obter expansões de acrónimos automaticamente e limpar os dados com menos esforço do utilizador.

Palavras-chave: Limpeza de dados; Aprendizagem de máquina; Expansão de acrónimos; Desambiguação de acrónimos; Envolvimento do utilizador.

Abstract

In recent years, due to fast data spreading and low cheap sensors, data has been created in a very fast pace. Data is continuously produced, scrutinized and increasingly used to make important decisions. Automated recommendations and decision-making based on these data can lead to enormous societal benefits. Conversely, the existence of large amounts of data increases the probability of occurring data quality problems with negative impact on data-based decisions. Data quality problems can be errors, missing values, values with doubtful meaning, duplicates and inconsistencies. A data cleaning process plays an important role in correcting these problems. Therefore, in this thesis, we address two main subjects: (i) acronym expansion that expands acronyms found in text to improve readability and (ii) support for the user involvement during a data cleaning process, to reduce the effort to effectively produce clean data.

The existence of acronyms with no expansion in text is considered a data quality problem, and it has been lacking attention from the data cleaning community. In fact, there is a lack of an acronym expansion software system available that can automatically find the expansions for the acronyms found in textual documents, and that is properly evaluated. Furthermore, in cases where an acronym has more than one expansion available, the criterion usually applied to select the right expansion is the cosine similarity between a representation of the document containing the acronym and a representation of each document containing an expansion. We claim that selecting the right expansion can be improved with other term and document representations and machine learning techniques that replace the cosine similarity.

A data cleaning process is usually an iterative process because it may need to be repeatedly executed and refined in order to be able to produce the highest possible data quality. Moreover, due to the specificity of some data quality problems and the limitation of data quality rules to cover all data cleaning problems, often a user has to be actively involved in the execution of a data cleaning program by manually repairing data. However, there is no framework that supports the user involvement in such iterative data cleaning process. Moreover, tools used for data cleaning that somehow involve the user in the process have not been evaluated with real users to access the user effort when designing data cleaning programs and manually repairing data.

In this work, we contribute with new approaches that provide an effective and effortless data cleaning process. In particular, we proposed: (i) an extensible end-to-end acronym expander system with novel acronym and expansion extraction and acronym disambiguation approaches together with three benchmarks to evaluate the performance of end-to-end systems and approaches for acronym and expansion extraction, and acronym disambiguation; and (ii) an improved data cleaning framework with support for user involvement during an iterative data cleaning process and conducted an experimental comparison of tools used for data cleaning with real and simulated users.

Finally, we evaluated our contributions. Specifically, the acronym expander system best pipeline scored 54.97% of F1-measure for expanding acronyms while the only related work system, MadDog, scored 32.93%. For the data cleaning framework, based on the answers to user questionnaires and

the user effort measured, it helps to reduce the user effort, scoring higher than the other software tools used for comparison (Pentaho Data Integration and OpenRefine). For instance, expert users on the Technology Acceptance Model (TAM) satisfaction questionnaire gave on average 30 in 42 points for ease of use and usefulness and on average did 95 clicks less when using our data cleaning framework to manually clean data. Thus, we conclude that the acronym expander system and the data cleaning framework provide new effective ways to obtain acronym expansions automatically and to clean data with less user effort.

Keywords: Data Cleaning; Machine Learning; Acronym Expansion; Acronym Disambiguation; User Involvement.

Contents

List of Tables

List of Figures

List of Acronyms

Acc	Accuracy
AcX	Acronym eXpander
ASCII	American Standard Code for Information Interchange
AVG	Average
BERT	Bidirectional Encoder Representations from Transformers
BFS	Breadth First Search
BPA	Big Publication Authors
CCV	Classic Context Vector
CFDs	Conditional Functional Dependencies
CIDs	Conditional Inclusion Dependencies
CL	Childhood Locations
Cossim	Cosine similarity
CPU	Central Processing Unit
CRFs	Conditional Random Fields
CS	Computer Science
Wiki	Wikipedia
DCV	Document Context Vector
DFS	Depth First Search
egds	equality generating dependencies
ETL	extract, transform, load
FDs	Functional Dependencies
GB	GigaBytes

GDR Guided Data Repair

GPU graphics processing unit

GUI Graphical User Interface

HTML HyperText Markup Language

IAA InterAnnotator Agreement

INDs Inclusion Dependencies

LDA Latent Dirichlet Allocation

LF Link Follower

LNCRF Latent-state Neural Conditional Random Fields

LR Logistic Regression

LSA Latent Semantic Analysis

MadDog-sys MadDog System

MadDog-in MadDog in-expansion

MASI Measuring Agreement on Set-valued Items

MDRs Manual Data Repairs

MDs Matching Dependencies

MEDLINE Medical Literature Analysis and Retrieval System Online

ML Machine Learning

MSH Medical Subject Headings

MTurk Mechanical Turk

NCRFs Neural Conditional Random Fields

NLP Natural Language Processing

NLTK Natural Language Toolkit

NOA Reuse of Open Access Images

PA Publication Authors

PDI Pentaho Data Integration

PK Primary Keys

PLSA Probabilistic Latent Semantic Analysis

QCs Quality Constraints

RAM Random Access Memory

RDBMS Relational DataBase Management System

RF Random Forests

SBE Surrounding Based Embedding

SBERT Sentence Bidirectional Encoder Representations from Transformers

SciDr-in SciDr in-expansion

SH Schwartz and Hearst

SQL Structured Query Language

SQLU Structured Query Language Update

SVMs Support Vector Machines

TAM Technology Acceptance Model

TBE TF-IDF Based Embedding

TF-IDF Term Frequency–Inverse Document Frequency

UAD Unsupervised Abbreviation Disambiguation

URL Uniform Resource Locator

VLDB Very Large Data Bases

Chapter 1

Introduction

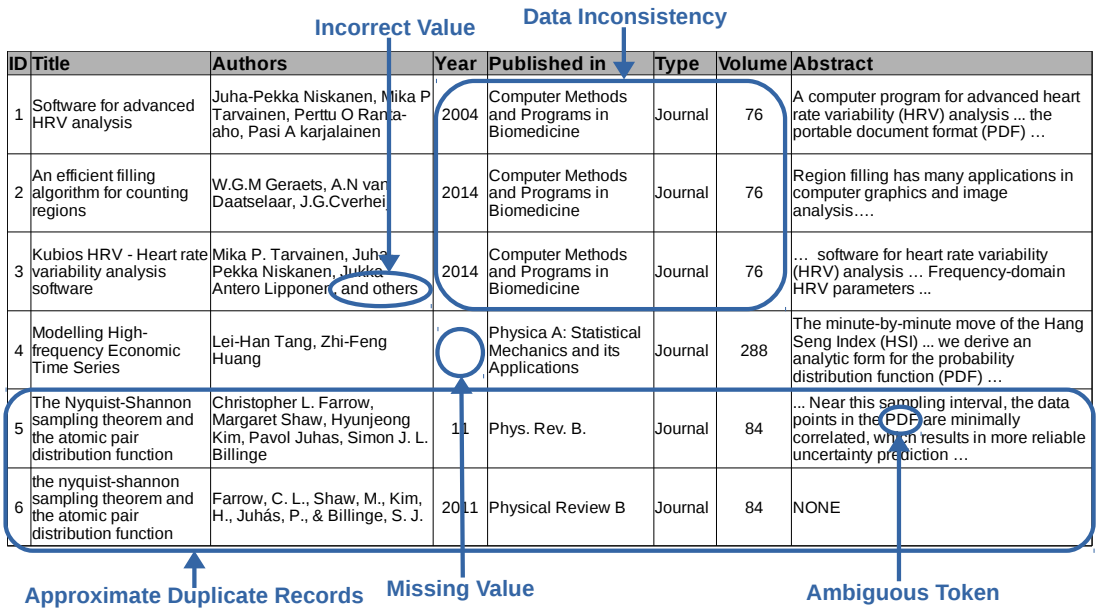
In this thesis, we propose new data cleaning solutions that are able to automatically improve quality and reduce the user involvement in a data cleaning process. In particular, we identify open problems in data cleaning regarding insufficient acronym expansion solutions and integration of user involvement during a data cleaning process in data cleaning frameworks. This chapter is organized as follows: in Section 1.1, we introduce data quality and its problems; in Section 1.2, we describe data cleaning and different approaches to produce clean data; in Section 1.3, we present the contributions of this thesis; and, finally, in Section 1.4 we present the outline of the rest of the document.

1.1 Data Quality

Nowadays, data is an essential asset of any business or public organization to support decision through data analysis or automatic recommendations. Moreover, data is the basis of the main products of Information Technology companies like Google, Facebook, Spotify, and Amazon. In order to guarantee successful decisions, data quality is essential, otherwise we end up with the known situation: "garbage in garbage out". In this context, garbage consists of data quality problems that results from a variety of situations, from human error and sensor malfunctions to data integration scenarios.

Poor data quality greatly impacts data-based decisions. The world's top companies have more than 25% of its critical data dirty [106]. The existence of dirty data may have disastrous consequences. For instance, wrong Electronic Health Records can lead to bad assumptions about a certain treatment. McKinsey estimates savings of 300 billion dollars every year in the US health care [67] by properly analyzing and correcting health data. Globally, Gartner measures the average impact of data quality in organizations as $9.7 million per year [105]. Moreover, heterogeneous data grows at an exponential rate, a phenomenon known as Big Data [65], which increases the probability of occurring data errors and data inconsistencies, thus, making data not reliable for analysis.

Data quality problems may be of different types [73]. Single value data quality problems enclose, for instance, missing values, incorrect values (e.g., syntax violations, misspelled errors, domain constraint violations), and ambiguous values (e.g., acronyms with no expansion). When considering a set of



ID	Title	Authors	Year	Published in	Type	Volume	Abstract
1	Software for advanced HRV analysis	Juha-Pekka Niskanen, Mika P Tarvainen, Perttu O Ranta-aho, Pasi A karjalainen	2004	Computer Methods and Programs in Biomedicine	Journal	76	A computer program for advanced heart rate variability (HRV) analysis ... the portable document format (PDF) …
2	An efficient filling algorithm for counting regions	W.G.M Geraets, A.N van Daatselaar, J.G.Cverhei	2014	Computer Methods and Programs in Biomedicine	Journal	76	Region filling has many applications in computer graphics and image analysis….
3	Kubios HRV - Heart rate variability analysis software	Mika P. Tarvainen, Juha-Pekka Niskanen, Jukka-Antero Lipponen, and others	2014	Computer Methods and Programs in Biomedicine	Journal	76	… software for heart rate variability (HRV) analysis … Frequency-domain HRV parameters ...
4	Modelling High-frequency Economic Time Series	Lei-Han Tang, Zhi-Feng Huang		Physica A: Statistical Mechanics and its Applications	Journal	288	The minute-by-minute move of the Hang Seng Index (HSI) ... we derive an analytic form for the probability distribution function (PDF) …
5	The Nyquist-Shannon sampling theorem and the atomic pair distribution function	Christopher L. Farrow, Margaret Shaw, Hyunjeong Kim, Pavol Juhas, Simon J. L. Billinge	11	Phys. Rev. B.	Journal	84	... Near this sampling interval, the data points in the PDF are minimally correlated, which results in more reliable uncertainty prediction …
6	the nyquist-shannon sampling theorem and the atomic pair distribution function	Farrow, C. L., Shaw, M., Kim, H., Juhás, P., & Billinge, S. J.	2011	Physical Review B	Journal	84	NONE

Figure 1.1: Example of data quality problems in a publications table.

records and/or attributes, we can find violations of domain constraints (e.g., a publication published in year 3000), inconsistencies (e.g., two publications with the same journal name and volume number that are not published in the same year), and approximate duplicates (e.g., two records referring to the same person).

Example 1.1.1. Consider Figure 1.1 that represents a table containing data about publications. Some quality problems can be observed: (i) the incorrect value “and others” in the list of authors for publication record 3, (ii) the data inconsistency that involves records 1, 2 and 3 given that they do not satisfy the condition that journal publications published in the same journal and volume must have the same publication year, (iii) a missing value in the year field of publication 4, (iv) the occurrence of the acronym PDF in the abstract field of publication 5, with no expansion and thus consisting on an ambiguous token, and (v) approximate duplicate records 5 and 6 referring to the same publication.

1.2 Data Cleaning

Data cleaning is today acclaimed as a central feature of data analysis and management tools. Data cleaning aims at converting source data into target data without errors, missing values, ambiguities, duplicates and inconsistencies [89]. It is thus of paramount importance to execute a data cleaning process to effectively eliminate data quality problems [95]. Customized data cleaning procedures can be implemented using a general purpose programming language, such as Java or Python, or a specialized software tool. Specialized commercial and research tools typically support different data operators that can be composed and form a transformation workflow to clean data. For instance, ETL (Extraction,

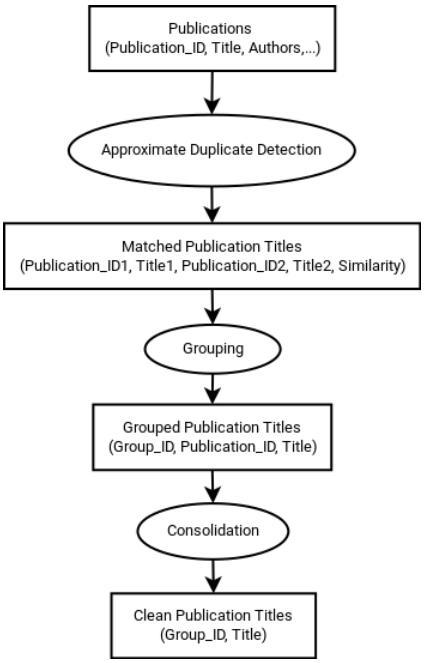


Figure 1.2: Part of a graph from a data transformation cleaning program.

Transformation and Loading) tools such as Informatica PowerCenter[1], IBM Data Integration[2], Talend[3], and Pentaho Data Integration (PDI)[4] provide a panoply of data operators that can implement several data transformations. Data wrangling tools, such as Trifacta Wrangler[5] based on the research tool Data Wrangler [53], and OpenRefine[6], provide an interactive process to implement data transformations and to map data to a specific format through data operators. Finally, there are tools specifically designed for data cleaning purposes such as the commercial tool Trillium Quality[7] and the research tools Febrl [19], LLUNATIC [39], NADEEF [26], and Ajax [36].

A data cleaning program is typically modeled by a program designer as: (a) a graph of data transformations [36] or (b) a set of data quality rules [34].

In the approach (a), a data cleaning program is modeled as a graph of data cleaning transformations whose execution produces cleaned data. By providing distinct and configurable data operators that can be composed to implement those data transformations, data cleaning programs based on graphs of data transformations can clean data inconsistencies, domain/syntax constraints, approximate duplicate records/attributes, missing values, and incorrect data if the correct values are available in a reliable data source, which is integrated in the data cleaning process.

Example 1.2.1. In Figure 1.2, we present part of a short data cleaning program based on data transformations. In this graph, ellipses represent high-level data transformations, rectangles represent data tables, and arrows represent data flows. This data cleaning program aims at identifying and consolidating approximate duplicate scientific publications similar to records 5 and 6 identified in Figure 1.1.

[1] https://www.informatica.com/products/data-integration/powercenter.html
[2] https://www.ibm.com/analytics/us/en/technology/data-integration/
[3] https://www.talend.com/
[4] http://community.pentaho.com/projects/data-integration/
[5] https://www.trifacta.com/
[6] http://openrefine.org/
[7] https://www.trilliumsoftware.com/products/tss/data-quality

First, the data cleaning program executes an *Approximate Duplicate Detection* transformation that computes the similarity among all possible publication title pairs and filters out the non-similar pairs of titles. Second, it applies a *Grouping* transformation to similar pairs of titles which assigns a *Group_ID* to each publication title using a transitive closure algorithm, thus guaranteeing that similar publications share the same *Group_ID*. Third, it applies a *Consolidation* transformation to find the best representative publication title for each group of publications that share the same *Group_ID*.

In the second type of data cleaning programs (b), *data quality rules* express conditions that data must satisfy to be considered of good quality. If data does not satisfy a rule, then a violation occurs and a data repair that corrects the data must be found. Data repairs are typically selected from the set of possible data repairs using heuristics. In the literature, several formalisms were proposed to express data quality rules. For example, the authors of [13] proposed Conditional Functional Dependencies (CFDs), an extension to Functional Dependencies [31].

By exploring different types of formalisms to specify data quality rules, these rule-based data cleaning programs can identify and possibly lead to the resolution of data inconsistencies, domain constraints, approximate duplicate records, and missing values and incorrect data if the correct values are available in a reliable data source.

Example 1.2.2. To detect the data inconsistency that involves records 1, 2 and 3 in Figure 1.1, we can use a CFD that represents the rule: all publications whose *Type* value is "Journal" and have the same *Published In* and *Volume* values, must have the same year value. Translating to CFD notation, this would correspond to: `Type[Journal], Published In, Volume ->Year`. In this situation, data cleaning consist in generating, selecting, and applying a data repair that modifies the publications records to satisfy the CFD. Possible data repairs are: (i) change the Year of tuple 1 to 2014 or (ii) change the Volume value of record 1 to a different value (e.g. 50).

It is worth noting that commercial data cleaning tools typically support data cleaning graphs [7] which are useful for users to control the data flow [55]. Nevertheless, the majority of recent research contributions rely on complex data quality rules and on the expensive generation of data repairs [55].

The data quality problem of ambiguous values, in particular the use of acronyms with no expansion, has not been considered by the data cleaning literature so far. In Section 1.2.1 we detail the acronym expansion that is a particular data quality problem. In Section 1.2.2 we explain the advantages and challenges for user intervention during a data cleaning process.

1.2.1 Acronym Expansion

Acronym expansion is the problem that finds an expansion for a given acronym in a textual document (e.g., "PDF" in Figure 1.1). Acronyms are used in all kinds of textual documents and articles. They are useful to shorten forms of a word or phrase; however, the lack of an acronym expansion in the text can become a communication barrier with non-expert persons. This problem is arduous when the same acronym is used across different areas thus leading to ambiguities due to many possible expansions,

e.g., “PDF” may mean “Portable Document Format”, “Probability Distribution Function”, or “Probability Density Function”.

Intuitively, an approach to acronym expansion comprises the following two steps: (i) Extraction of both acronym and (when present) its expansion within a text. For example, if a given text has “we derive an analytic form for the probability distribution function (PDF)” then PDF would be the acronym and probability distribution function would be the expansion. We call this *in-expansion* because this can be done for a particular article on its own. (ii) In the case that an acronym is not expanded in a text, *out-expansion* chooses an expansion from a previously large parsed corpus (training corpus) like Wikipedia based on some notion of article domain similarity.

Acronym and in-expansion extraction obtains the acronyms and associated expansions from a given text. These expansions can not only be used to expand acronyms within the document, but also to expand acronyms in other documents where the expansion is not available in text. Often, this is achieved by using rule-based techniques [97][96][110] that identify acronyms and in-expansions through patterns e.g., an expansion followed by an acronym between parenthesis, for example *probability distribution function (PDF)*. Some recent works [117][41][63][50][99] explore the use of machine learning to extract acronyms and expansions from text that are not bounded to particular patterns but still need training data. These techniques were not evaluated against each other, specifically they often make use of datasets from different domains or different annotated versions of the same dataset, making it hard to identify the current state-of-the-art.

The bottleneck in acronym expansion is the selection of documents based on domain similarity. Document selection is usually achieved by representing each document by a context that accurately maps it into the most important domain concepts. Thus, context extraction plays an important role on finding the right out-expansion. To extract context, existing works have used terms (e.g., acronyms or expansions) or textual document representation techniques, such as Doc2Vec [58] used in [107], and a combination of Word2Vec [68] outputs in [61], but they did not explore other term or document representation techniques such as Term Frequency–Inverse Document Frequency (TF-IDF) or Latent Dirichlet Allocation (LDA). Moreover, in [107], the authors did not use a global Doc2Vec model trained on the whole corpus, instead they used a local model for each acronym trained on just the documents that contains at least an expansion for the acronym. Although assigning an expansion to an acronym in text can be seen as a Machine Learning (ML) classification problem, finding the most similar document using context has been typically addressed with heuristics.

Recently, works in out-expansion make use of neural networks: Maddog [110] proposes a sequential model to encode context in sentences followed by a feedforward network to classify the input sentence with an expansion; and SciDr [99] formulates the problem as a span prediction task, where given a list of expansions concatenated with a sentence as input, it uses the pre-trained language model SciBERT [9] retrained in the dataset sentences to predict the span, i.e., start and end word indices corresponding to the predicted expansion.

Maddog [110] also proposed a system for acronym expansion. However, the authors used fixed approaches for acronym and expansion identification, and for acronym disambiguation. The documents

used to build their dataset were also fixed, and only the resulting trained models were shared.

So far, there is no end-to-end off-the-shelf acronym expansion system that receives as input text acronyms and outputs a list of acronyms with their expansions. Moreover, there is lack of: (i) a framework where new approaches can easily be plugged in such system and (ii) complete benchmark studies that include: evaluations for acronym and in-expansion extraction, and out-expansion tasks, that makes use of distinct datasets from different domains (e.g., biomedical, computer science), and that also evaluates the end-to-end systems using an end-to-end benchmark with a corresponding real annotated dataset.

As a final note, an acronym expander system can be introduced in the data cleaning pipeline as a data transformation, i.e., replacing acronyms in text fields by the correct expansion like in the paper abstracts example in Figure 1.1. Such approach can help for instance to find more accurately approximate duplicate records or even to impute values into missing/incorrect cells, e.g., by having expansions instead of acronym in the abstract texts, it becomes clear to find the main subjects of such texts and compare them by similarity.

1.2.2 Human Intervention in a Data Cleaning Process

Often, the execution of a data cleaning program is insufficient to clean all the data quality problems present in real databases. This happens because: (i) the data quality rules or transformations available cannot cover all the cleaning criteria required to generate clean data, (ii) generated data repairs or transformations may not clean all the data, leaving some dirtiness, or (iii) in case of machine learning-based data cleaning, the available clean data may not be sufficient to infer the correct values (e.g., infer the name of a publication). For instance, missing and incorrect values (e.g., missing year) cannot be cleaned by data transformations nor generated data repairs without using an external data source that contains the correct values. Another example is when detecting approximate duplicates. In practice, there is no approximate duplicate criteria (i.e., similarity function and threshold) that perfectly identifies records that refer to the same entity in the real world. The incorrect identification of approximate duplicate records results in false positives or false negatives thus leaving some data dirtiness.

In data cleaning processes, users with domain knowledge typically have to manually repair some data items in order to guarantee the highest quality levels of the resulting data. In these cases, enabling a user to manually clean a subset of data during the execution of a data cleaning process (i.e., including the human in the process) is important. For instance, the authors of [37] proposed to enhance Ajax by modeling a data cleaning process as a data flow graph that encloses manual data repairs to allow the user to manually clean quality constraint violations. In LUNATIC, user feedback is incorporated into the data cleaning processes, through the notion of lluns. LLUNATIC requires user feedback when a repair conflict is found (i.e., when it cannot automatically select a preferred value). If the user modifies the cleaning program (e.g., refines the quality constraints), the cleaning process starts from scratch creating distinct repairs and lluns.

Currently, there is no data cleaning framework that supports, in a principled way, the user involvement during an iterative execution of such a process. Furthermore, no evaluation of data cleaning tools (ETL

and data wrangling) with real users to measure the user effort when designing data cleaning programs and manually correcting data has been performed so far. In this thesis, we focus on providing a solution for the above mentioned problems.

1.3 Main Contributions

Taking into consideration the current gaps in research regarding data cleaning, identified in Section 1.2, in this work, the following main contributions were produced:

Acronym Expansion: a comprehensive end-to-end system, framework, and benchmark for acronym expansion in text documents. For acronym and in-expansion extraction, this system uses efficient approaches from related work [97][99][110]. For out-expansion, apart from the support for numerous related work approaches [61][107][99][110], we proposed and evaluated new out-expansion approaches: (i) term and document representation techniques from topic modeling and vector representation to extract the document context, and (ii) machine learning techniques to use the context and select an expansion. The different approaches were evaluated by benchmarks specially designed for acronym and in-expansion extraction and out-expansion. We also created the first end-to-end benchmark for such systems that includes a new annotated dataset and the only one to evaluate acronym expansion when expansions exist in text or not. This system also includes a user interface to allow the user to upload documents and to visualize the expansions of acronyms in text.

Support for User Involvement: an extension of the data cleaning framework, called Cleenex[8] that is based on the graph-based data cleaning framework called Ajax. In Cleenex, a data cleaning graph is equipped with data quality constraints and manual data repairs that provide support for user involvement to manually clean data as in [37]. We extended Cleenex into a stable release that supports the user involvement in the iterative execution of a data cleaning program. Furthermore, we performed an exhaustively evaluation of the proposed framework using real users and to perform a comparison with existing tools used for data cleaning in what concerns the required user effort.

1.4 Document Outline

The remaining of this document is organized as follows:

Chapter 2: presents the fundamental concepts of data cleaning, and term and document representation to obtain context from documents.

Chapter 3: summarizes the most important related work about acronym and in-expansion extraction, out-expansion, and user involvement in cleaning data.

[8] http://web.tecnico.ulisboa.pt/~ist164790/cleenex/

Chapter 4: describes the details of the contributions of this thesis regarding acronym expansion.

Chapter 5: describes the details of the contributions of this thesis to efficiently support the user involvement in data cleaning programs.

Chapter 6: summarizes the main conclusions of this thesis and describes ideas for future work.

Chapter 2

Fundamental Concepts

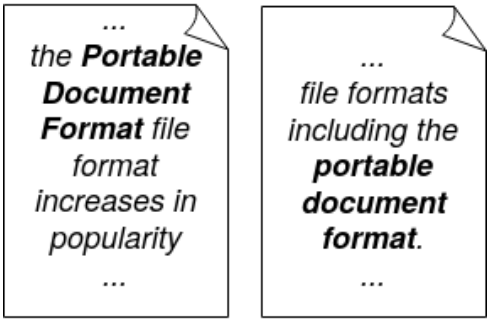
In this chapter, we present the main concepts of data cleaning, and term and document representation for context extraction used for acronym out-expansion. In Section 2.1, we explain the main techniques used to produce document or term representations. Our acronym expander system can use some of those representations (e.g., LDA, Doc2Vec) to out-expand the acronyms not found in text. Section 2.2 explains the data cleaning process and the main approaches. Specifically, this section starts with a general overview of the typical data cleaning process and then focus on the two main approaches to model a data cleaning process: data transformations and data quality rules.

2.1 Term and Document Representations

In this section, we explain well established and recent representation techniques for terms (i.e., meaningful words/tokens) and documents. Representation techniques assign to an object (term or document) a representation that is semantically more meaningful than byte representations like Unicode and ASCII, and the one-hot encoding (i.e., a vector whose positions corresponds to the co-occurrence of a distinct word with the object) used as features in Machine Learning approaches. Topics represent text (e.g., a document) by a reduced set of the most representative terms (i.e., topics). Thus, documents whose most representative terms are common are semantically related. Representation techniques that outputs a set of terms for a document are referred as topic modeling. Other representations called embeddings represent an object (term or document) by a continuous vector of real numbers in a dimensional space. Embedding techniques map an object encoded in an one-hot representation, a very sparse and high dimensional vector of binary values, into a very dense and lower dimensional vector of real values (i.e., embedding). These techniques try to capture semantic information into the embeddings, intuitively, the distance between vectors carry a semantic meaning (e.g., synonym words).

Representation techniques are useful for disambiguation problems (e.g., Word Sense Disambiguation and out-expansion) where the goal is to select the right definition for an ambiguous term taking into account semantic information.

Documents containing the *Portable Document Format* term



Classic Context Vector for the *Portable Document Format* term

Positions	the	file	format	increases	in	popularity	formats	including
Values	2	2	1	1	1	1	1	1

Figure 2.1: Classic context vector example for *portable document format* term.

In the following subsections, we detail the main techniques to semantically represent terms or documents: in Section 2.1.1 the classic version of context vectors; in Section 2.1.2, we present Term Frequency–Inverse Document Frequency (TF-IDF), a statistical technique that combines statistics about the occurrence of terms in a document and across the corpus; in Section 2.1.3, Latent Semantic Analysis (LSA), a topic modeling technique, that assigns topics to documents; presented in Section 2.1.4, Latent Dirichlet Allocation (LDA) a topic modeling technique like LSA that uses probabilistic models; in Section 2.1.5 Word2Vec, a word embedding technique that assigns semantic vectors to words; in Section 2.1.6, Doc2Vec, a document embedding technique based on Word2Vec that automatically assigns vectors to documents; in Section 2.1.7, BERT, a transformer language model that can generate word embeddings; and, finally, in Section 2.1.8, Sentence-BERT, a sentence embedding technique that uses BERT to assign vectors to sentences.

2.1.1 Classic Context Vectors

The context vector technique is an unsupervised method used as a baseline in Word Sense Disambiguation problems [2] and also in out-expansion problems [85][61]. We denote it as classic to distinguish it from variants or other techniques that also provide vectors to contexts.

A *Context vector* represents a term (e.g, an acronym or expansion) by a vector based on the words that co-occur with the term in each document of the corpus. Thus, a context vector is a sparse vector where each position corresponds to a word in the corpus (except the terms). In the classic approach, the value at each vector position corresponds to the number of co-occurrences of the term and the co-occurring words in all documents. However, other variants are possible where the value is just 1 or 0 to indicate, respectively, the existence of word co-occurrences (or the co-occurrence relative frequency) or their absence regarding the term. Figure 2.1 presents an example of a context vector for *portable*

document format term using two documents.

In disambiguation problems (either word sense disambiguation or out-expansion), the ambiguous term (e.g., acronym) in a particular document leads to a context vector which contains only the number of word occurrences in that document. Each possible definition (e.g., expansion) for the ambiguous term will have a context vector as explained above. To select an expansion to an acronym, we usually compare the context vectors using the cosine similarity to find the most similar context vectors.

2.1.2 TF-IDF

The Term Frequency–Inverse Document Frequency (tf-idf or TF-IDF) [102] is a statistical technique that is used to obtain the most important terms in a particular document (e.g., topics). An important term for a particular document can be defined by two factors: (i) its frequency (TF) within the document, intuitively terms that appear more times in a document tend to be more important to define context and (ii) its exclusivity regarding other documents (IDF), so if a term appears in a large set of documents it might not be a good document representation topic. One example are conjunction words (e.g., so, and, or) that appear often within a document but carry little topical value. TF-IDF takes into account both of these factors by multiplying the term frequency (i) and the inverse document frequency (ii).

The term frequency $tf(t,d)$ for a term in a vocabulary $t \in T$ and for a document in a corpus $d \in D$ is basically the relative frequency of t in d, i.e., the number of term t occurrences in d divided by the total number of terms in d. The inverse document frequency $idf(t,D)$ for a term t in a set of documents D is the logarithm of the inverted frequency of term t in documents D, $idf(t,D) = log(\frac{\#D}{\text{\# documents in } D \text{ containing term } t})$. The final TF-IDF value $tf\text{-}idf(t,d,D)$ for a term t in a document $d \in D$ is obtained by multiplying the term frequency by the inverse document frequency, $tf\text{-}idf(t,d) = tf(t,d).idf(t,D)$

TF-IDF values can be used to rank the terms in a particular document, i.e., by selecting the top N terms we can obtain a document representation which is composed by the N terms (e.g., topics) for that document [66].

2.1.3 LSA

Latent Semantic Analysis (LSA) [29] is a topic modeling technique and a statistical technique used to obtain the main important terms (i.e., topics) that represent documents. The intuition behind LSA is that terms with similar semantics (e.g., synonyms) will occur in similar texts. In information retrieval literature, LSA is known as Latent Semantic Indexing [66].

LSA starts with a document-term matrix M, where each term $t \in T$ is represented by a row and each document $d \in D$ is represented by a column. This matrix M usually contains at each position $i \in \{1,...,|D|\}, j \in \{1,...,|T|\} : m_{i,j}$ the absolute frequency of a term t in a document d, i.e., number of occurrences of t in d. However, other matrices can be constructed for instance using the TF-IDF values for a term in a document as explained in Section 2.1.2.

Using a matrix $M_{T\text{x}D}$, LSA applies the Singular Value Decomposition (SVD) [103], a technique from

linear algebra used to factorize a matrix $M_{T\text{x}D}$ into three matrices $M_{T\text{x}D} = U_{T\text{x}T}S_{T\text{x}D}V^T_{D\text{x}D}$ where S is a diagonal matrix of singular values, U contains the eigenvectors of MM^T and V contains the eigenvectors of M^TM. Both U and V are orthogonal matrices. The resulting matrices U and V correspond to term vectors and document vectors respectively. Note that, each vector position is a hidden term with no representation in a vocabulary called latent term and there are as many latent terms as terms $t \in T$. We can then reduce these vector dimensions by accepting only the k most important latent terms, i.e., the latent terms with the highest variance regarding other terms. We can do this by keeping only the first k columns in U and V (corresponding to the k highest singular values in S), i.e., dimensionality reduction. Note that the resulting matrices (k dimensional term and document vectors) from LSA can be compared (e.g., through cosine similarity) to find similar documents, similar terms or similar documents to a set of terms. Nevertheless, the vector positions (either terms or documents) are not interpretable by humans as they do not map to a single document or term but to latent terms, i.e., abstract terms that do not correspond to a word in a human language.

2.1.4 LDA

Latent Dirichlet Allocation (LDA) [12] is a popular topic modeling and an unsupervised learning technique based on a generative statistical approach that models the probability of a term occurring in a document.

LDA is also known as a Bayesian approach to Probabilistic Latent Semantic Analysis (PLSA) [46] (Probabilistic Latent Semantic Indexing (PLSI) in information retrieval literature) a technique based on LSA whose intuition is to model the probabilities of a set of terms t belonging to a topic c and the probability of a topic c describing a document d [40]. Thus, PLSA models the probability of a term t occurring in a document d as a mixture of conditionally independent multinomial distributions, $P(t,d) = P(d)\sum_{c\in C=\{c1,\ldots,c_K\}} P(c|d)P(t|c)$, where C are the topics with no representation in a vocabulary called latent topics, K is a hyperparameter (parameter provided by the user, as opposed to a learned parameter) representing the number of topics, $p(d)$ is the probability of the occurrence of a document d, $P(c|d)$ is the conditional probability of the occurrence of a topic c given a document d, and $P(t|c)$ is the conditional probability of the occurrence of a term t given a topic c. A topic here is defined by a set of terms. The parameters of the models are obtained using the Expected Maximization (EM) algorithm.

LDA is similar to PLSA but uses the Dirichlet distribution as a prior probability distribution (i.e., models the uncertainty for unseen values) for two types of probabilities: (i) the probabilities of a topic describing a document and (ii) the probabilities of a term being part of a topic. This leads to two hyperparameter vectors α and β which are the parameters of the Dirichlet distribution priors, α for a topic describing a document and β for a term belonging to a topic. Considering symmetric vectors, lower values of α promote fewer topics to represent a document, likewise, lower values of β promote fewer terms composing a topic. Other hyperparameters are the number of topics, and the number of passes through the corpus.

LDA generalizes better than PLSA because it uses Dirichlet priors that model uncertainty and so prevent overfitting [12]. Another advantage over PLSA is that the LDA model is interpretable by humans,

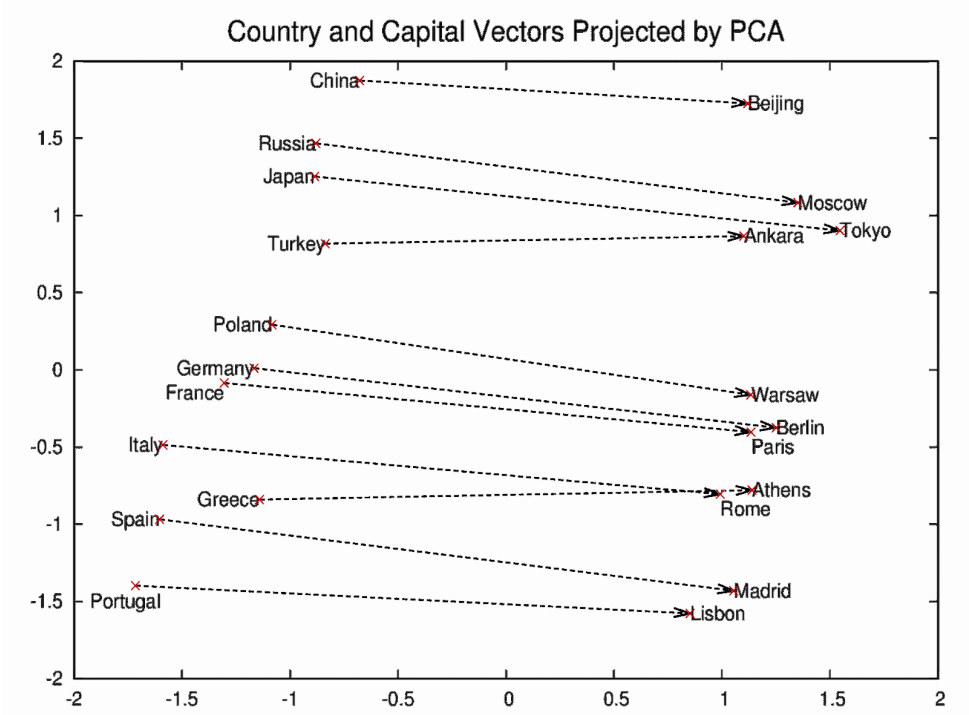


Figure 2.2: Countries and Capitals vectors. Modified from [69].

i.e., topics are composed by non latent terms (e.g., words) and documents are characterized by a set of topics, each with an associated percentage.

2.1.5 Word2Vec

Word2Vec [68] is an unsupervised learning technique that processes large collections of documents and assigns a vector, called a word vector, of some dimension N (i.e., embeddings) to each word. Vectors are assigned in such a way that words that appear in the same context have a high cosine similarity.

For example, consider Figure 2.2 which maps word vectors corresponding to names of countries and capitals into a 2-dimensional space [69]. We can observe that Spain and Portugal are close because they are often mentioned together in text because they resemble one another in terms of geography and culture. Lisbon and Madrid are close to each other because they are the capitals of Portugal and Spain.

Given these vector representations, another interesting Word2Vec analysis is possible. For example, if we want to know the capital of Portugal, we can perform operations using the word vectors, such as Portugal + Spain - Madrid which would result in a vector close to Lisbon. This is possible because the countries and capitals relationship in text is modeled by Word2Vec using similar distances between vectors from a country to vectors of its respective capital.

In order to generate the embeddings for each word, Word2Vec is trained using an example text divided into sentences where words are considered related if they often occur in a substring of a sentence. The training problem consists of finding the best set of embedding values for each word (i.e., Word2Vec model parameters) that, given a word or a set of words, predict the word or set of words that occur in

the same context [93].

As mentioned above, word vectors are usually compared using cosine similarity. Two word vectors having high cosine similarity are supposed to correspond to words having high semantic similarity (e.g., Portugal and Spain in Figure 2.2). Word vectors can also be used to describe a context (e.g., document). For that purpose, the common approach is that the vectors of the words in a particular context are summed together and divided by the number of words, leading to a context vector based on Word2Vec. For example, consider we have a document that just lists the countries names in Figure 2.2. Then, the context vector is computed by summing all countries vectors and then divide by the number of countries. The result would be a vector near Germany or France.

2.1.6 Doc2Vec

Doc2Vec [58] is a document embedding and an unsupervised learning technique that adds to Word2Vec the capability of automatically learn document (or paragraph) vectors. Given a list of words (e.g., a text document) as input, the output of Doc2Vec is a dense vector of real numbers (i.e., embedding).

Analogously to Word2Vec that assigns a vector to a word, Doc2Vec assigns a vector of N dimensions called a document vector to a document. The training problem consists of finding the best set of embedding values for each word and document (i.e., Doc2Vec model parameters) that, given a document, predicts the set of words in that document. For example, consider a document consisting on a list containing the countries in Figure 2.2. If the document is known to the Doc2Vec model (i.e., was in the training data) then we have a document embedding available, otherwise, a document embedding d is computed by finding the best values that maximize the prediction of the country names given d.

Instead of averaging word vectors to represent a particular document in Word2Vec, when using Doc2Vec, one can have a trained vector for each document in the corpus [25]. By comparing those document vectors through cosine similarity, we can infer semantic similar documents.

2.1.7 BERT

BERT [28], acronym for Bidirectional Encoder Representations from Transformers, is the most well known language model based on the transformer architecture. It revolutionized several Natural Language Processing (NLP) tasks by providing a language model trained in huge amounts of unlabeled text data that could be later fine-tuned for specific NLP tasks like Question and Answering (Q&A). Other successful models with similar architectures based on transformers followed like XLNet [116], RoBERTa [64] and MPNet [101]. These transformer-based models are deep neural networks composed by dense and attention layers. Most of the following BERT details are applicable to them.

The BERT language model is pre-trained on two tasks that can be performed in an unsupervised way without training data; they are: (i) predict a hidden word in text known as Masked Language Model and (ii) given two sentences, the probability of one following another in text called Next Sentence Prediction.

For training in the context of other tasks, BERT is usually leveraged with additional layers; for instance, a classification layer based on softmax can be added to label an input sentence or even a pair

of sentences.

The last layer output, before the additional classification layer, can also be seen as producing word embeddings where each word has a different embedding depending on the sentence. Importantly, homonym words will have a different embedding depending on context (like ”bank” where you store your money and ”bank” where you sit on). Combinations of word embedding vectors, like in Word2Vec [68], can be used to obtain a document embedding.

2.1.8 SBERT

Sentence BERT (SBERT) or sentence transformers is a Python library whose first model was introduced in [91] and that leverages transform models (not just BERT) to generate accurate sentence embeddings for sentence similarity tasks.

Given a pair of sentences $\{s_1, s_2\}$, SBERT models learn to encode each sentence $s \in \{s_1, s_2\}$ in such a way that the Cosine similarity of their embeddings reflects their true similarity. To do so, SBERT uses a Siamese neural network architecture composed by transform models that trains the model to predict if two sentences are similar or not.

On prediction time, the sentences' embeddings can just be compared using Cosine similarity. This is computationally more efficient as it runs the model per sentence and not per sentence pair.

Sentence embeddings can be used as document embeddings. They can also be merged if the document text exceeds the SBERT model input token limits.

2.2 Overview of a Data Cleaning Process

A typical data cleaning process aims at correcting the data quality problems in a database and it is part of a data quality process. Data quality problems can affect distinct data structures within the same data table: (i) a single value in a data table cell (structure that contains an attribute value in a particular record) such as missing and incorrect data; (ii) a set of values for a single attribute such as unique violation (e.g., two records with the same identifier); (iii) a set of attributes for a single record such as inconsistencies among attribute values, or (iv) a collections of multiple records and attributes such as approximate duplicate records. Data quality problems can also exist in the context of multiple relational tables, that may come from different databases, such as inconsistencies among attribute values, syntax inconsistency, or integrity violations (e.g., a foreign key value that does not exist as a primary key value). The typical data cleaning process only stops when the user is satisfied about the data quality achieved (i.e., the database meets the desired data quality level).

Data quality problems may be of different types [73]. Single value data quality problems enclose, for instance, missing values, incorrect values (e.g., syntax violations, misspelled errors, domain constraint violations), and values with doubtful meaning (e.g., acronyms with no expansion). When considering a set of records and/or attributes, we can find violations of domain constraints (e.g., a publication published in year 3000), inconsistencies (e.g., two publications with the same journal name and volume number

that are not published in the same year), and approximate duplicates (e.g., two records refer to the same person).

A data quality process usually starts with a data quality analysis phase which is the act of discovering data quality problems in a database such as data errors, data inconsistencies, and approximate duplicate records. Data quality analysis and data cleaning are obviously complementary and so, they can be offered by the same framework or commercial software. Moreover, in some tools, support for data quality analysis and data cleaning is intermixed because the system discovers and cleans data errors and inconsistencies using the same components.

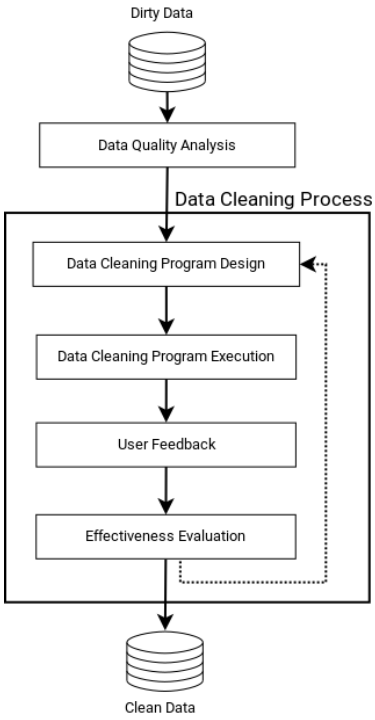


Figure 2.3: Data Quality Process

In Figure 2.3, we present a typical data cleaning process. Usually, this process is preceded by a *Data Quality Analysis* process that is intended to audit the data through data quality rules and statistical techniques. These techniques provide information about the data quality and thus allow the user to identify possible data quality problems. Then, the user should have sufficient information to proceed with the *Data Cleaning Program Design* (with data transformations or data quality rules) that can clean the data quality problems found so far. The next step is the *Data Cleaning Program Execution* followed by a step where the user can provide *User Feedback* by manually correcting instances of data quality problems not addressed by the automatic methods. Before declaring the data cleaned, in the *Effectiveness Evaluation* phase, the effectiveness of the data cleaning process is checked by measuring the data quality of a sample taken from the processed data. If the data is not sufficiently clean, the same process is reapplied to this data with a refined data cleaning program until the desired data quality is achieved.

In the next sections, we will present the two main types of data cleaning approaches. First, we analyze data transformations in Section 2.2.1 and then data quality rules in Section 2.2.2.

2.2.1 Data Transformations

In this section, we describe the data cleaning process based on data transformations. This method of cleaning data is the most used in companies. A data cleaning program constituted of data transformations is usually designed based on a drag-and-drop graphical user interface, identical to most user interfaces supported by ETL software, where data transformations are represented by boxes and the data flows by links between those boxes.

A data cleaning program based on data transformations is usually modeled as a graph of data transformations. The effective combination of data transformations within a graph and the proper tuning of each data transformation can transform dirty tables into clean tables.

Figure 2.4: Stage tables from transforming a table containing authors name by publication into a table containing similar author names.

(a) Initial table.

PUBLICATIONAUTHORS

Pub ID	Authors
1	Kanamori, Heiken
2	Kaanamori
3	Reeken, Kanamori
4	Knamoqri, Yothers

(b) List of author names.

AUTHORS

Author ID	Name
1	Kanamori
2	Heiken
3	Kaanamori
4	Reeken
5	Kanamori
6	Knamoqri
7	Yothers

(c) Pairs of similar author names.

SIMILARAUTHORS

Author ID 1	Name 1	Author ID 2	Name 2	Distance
1	Kanamori	3	Kaanamori	1
1	Kanamori	5	Kanamori	0
1	Kanamori	6	Knamoqri	2
2	*Heiken*	*4*	*Reeken*	*2*
3	Kaanamori	5	Kanamori	1
5	Kanamori	6	Knamoqri	2

Example 2.2.1. Consider that, using an extraction software, we have placed publication references in a database table, from which, we now want to obtain a list of approximate duplicate author name pairs. We start from an instance of this database table named *PublicationAuthors* as in Figure 2.4a that contains two attributes: (i) *Pub ID*, the publication identifier and (ii) *Authors*, containing the last names of each publication author separated by commas. The final expected result is the *SimilarAuthors* table, shown in Figure 2.4c, whose schema has 5 attributes: (i) *Author ID 1*, the identifier of the pair first author, (ii) *Name 1*, containing the last name of the pair first author, (iii) *Author ID 2*, the identifier of the pair second author, (iv) *Name 2*, containing the last name of the pair second author, and (v) a distance value between the two author's last names within the same pair. In order to obtain such table, we have to execute an Approximate Duplicate Detection operation composed by the following two high level data transformations:

Splitting: Splits each field value into a different row using a character or pattern as a separator. In

the example, Splitting extracts the individual author names from the string field *PublicationAuthors.Authors* using a comma as a separator (Figure 2.4a to Figure 2.4b).

Approximate Duplicate Detection: Computes a distance value using a distance function (e.g., the edit-distance [60] function) for all possible pairs of value for the input field *Authors.Name* (in Figure 2.4b). Then, it filters the rows using a given threshold value, e.g., distance values below 3 (Figure 2.4b to Figure 2.4c).

Finally, in this example, to emphasize the need for the user to *manually correct instances* of data quality problems not addressed by the automatic methods. Note that in Figure 2.4c, the names *Heiken* and *Reeken* (tuple in Italic) do not refer to the same real author and so that tuple needs to be removed.

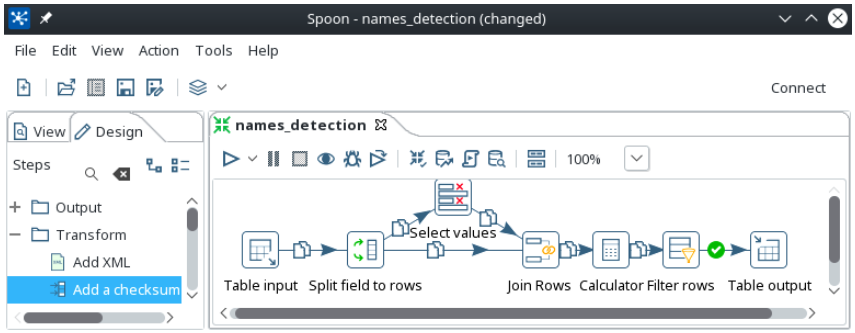


Figure 2.5: Pentaho Data Integration Graphical User Interface and implementation of a data cleaning program.

Usually, a program designer uses a specialized tool to implement those data transformations. Possibilities are ETL tools that support data cleaning transformations as Pentaho Data Integration (PDI), Talend, or Informatica PowerCenter, or data quality tools such as Trillium. As an example, we will focus on PDI which is the ETL tool of the Hitachi Vantara Company that provides an open-source version. PDI offers more than 200 data operators, named *steps*, that allow the user to design and execute data cleaning processes, besides ETL processes. PDI can be used to design and perform data cleaning tasks, since it provides data transformation steps typically used in data cleaning programs. A data cleaning program in PDI that corresponds to the Splitting and Approximate Duplicate Detection transformations of Example 2.2.1 is shown in Figure 2.5. These transformations are obtained through combinations of the following steps: (i) a *Table input step* obtains the table *PublicationAuthors* from a database and injects each tuple into the data flow; (ii) a *Split field to rows step*, following the *Table Input Step*, extracts, into the data flow, the author names for each tuple of table *PublicationAuthors*; (iii) a *Select values step* is used to duplicate the data flow output of the *Split field to rows step* and to rename the attributes because the next step cannot receive as input two data flows with equal schema; (iv) a *Join rows step* computes a cartesian product between two data flows: one produced by the *Split field to rows step* and the other produced by the *Select values step*, thus generating possible pairs of names; (v) a *Calculator step* accepts as input the pairs of names produced by the *Join rows step*, and computes the edit-distance between those names; (vi) a *Filter rows step* outputs only the pairs of names from the *Calculator step* whose distance is below 3; and finally (vii) a *Table output step* is used to insert the data flow into a database

table.

In this section, we described a data cleaning process through a set of data transformation. In the next section, we will describe another type of approach to achieve clean data that consists on a set of data modifications to resolve the disagreement between data quality rules and data.

2.2.2 Data Quality Rules

Data quality rules express conditions that data must satisfy to be considered of good quality. When a rule is violated then a data quality problem occurs. Searching the data for such problems is not a trivial task, even more when complex rules are introduced. In this section, we describe the type of rules available in data quality research for data quality analysis and the data cleaning process based on such rules.

Conceptually, the first rules implemented in data management software (e.g., RDBMS - Relational Database Management Systems) were *integrity constraints*. Integrity constraints describe conditions among data values, for instance, a normal employee cannot earn more than the director of a company. Integrity rules were first used for other contexts of data management software, specially in ETL tools, however users also used and use them to express data quality requirements. Two software tools that provide rules for data cleaning purposes are IntelliClean [59] that uses business rules to identify data inconsistencies and missing data, and TransScm [70] which explores rules for approximate duplicate detection. *Functional Dependencies* (FDs) [31] are integrity rules that check if, for any set of tuples whose values of a set of user specified attributes X are equal (e.g., same Zipcode), the values of those tuples in another specified set of attributes Y (e.g., same City) are equal. An FD is represented as X `->`Y. As an example, consider a table Customers storing person records that contain location related attributes such as City, Zipcode, and Country. We may state that if two or more records share the same zip code then they must have the same city value. The corresponding FD is: `Zipcode -> City`. *Conditional Functional Dependencies* (CFDs) [13] are extensions to FDs that enable to use value conditions for attributes. Consider that the FD previously defined was only valid in the United States of America, then we can specify a CFD that expresses that condition as follows: `Country[United States of America], Zipcode -> City`. Other formalisms are: *Inclusion Dependencies* (INDs), which express conditions that data records must satisfy across relations (e.g., a foreign key in a database); *Conditional Inclusion Dependencies* (CIDs) [14], which enable to express conditions over INDs (like the CFDs did with FDs); and *Matching Dependencies* (MDs) [35], which are dependencies designed for approximate duplicate detection. Finally, *equality generating dependencies* (egds) [8] consists on a formalism that standardize dependencies by proposing a syntax based on logic that encloses most of the data dependencies from the literature (e.g., CFDs, CIDs). In egds format, the CFD example presented before would correspond to: `Customers(Name1, Address1, City1, Zipcode, Country), Customers(Name2, Address2, City2, Zipcode, Country), Country=‘‘United States of America ’’ -> City1 = City2`.

As stated above, data must satisfy the conditions described by a set of data quality rules, otherwise a

violation occurs meaning that data quality problems are found. To resolve a violation, there are different alternatives to modify the data called *data repairs*. For example, consider the records 1, 2 and 3 in Figure 1.1 and the CFD `Type[Journal], Published In, Volume -> Year`, an obvious data repair is to change the Year of tuple 1 to 2014. Another valid data repair would be to change the volume value of record 1 to a different value such as 50. Choosing which data repair to apply is usually performed through heuristics [23][18][39][26].

A data cleaning tool whose objective is to discover and select data repairs to solve violations to specified data quality rules, such as LLUNATIC [39] and NADEEF [26], usually starts with a violation detection phase. Violation detection is the act of discovering data tuples that violate data quality rules designed by users. Violation detection finds data dirtiness, i.e., tuples that violate at least a rule called *violating tuples*. The second phase, candidates generation phase, typically executes a chase procedure to discover the possible data repairs to be applied to the database. Finally, a search phase occurs in order to select the best set of data repairs that produce clean data. Finding the best set of data repairs is not a trivial task because a data repair that modifies the data may also generate new violations to other rules. Another important issue to take into consideration is the definition of the best data repair set, which is usually defined heuristically as being the minimum number of modifications applied to the original data that can produce a database with no violations.

In this section, we described the different types of rules for data quality analysis in the context of data cleaning. Moreover, we described that, when a data quality rule is not satisfied, then a violation occurs and a repair must be applied to correct the database.

Chapter 3

Related Work

This chapter describes the most relevant work related to acronym expansion (in-expansion extraction, and out-expansion) and user involvement in repairing data. In Section 3.1, we describe the work about acronym and out-expansion extraction and in Section 3.2 the work about out-expansion. In Section 3.3, we describe the few end-to-end systems for acronym expansions. Finally, in Section 3.4, we focus on data cleaning literature that makes use of user feedback to repair data.

3.1 Acronym and In-expansion Extraction

In this section, we review work about acronym and in-expansion extraction that identify acronyms and corresponding expansions in text. In Section 3.1.1, we present a simple but efficient rule-based algorithm for acronym and in-expansion extraction. In Section 3.1.2, we report on work that proposes improvements over this simple algorithm. In Sections 3.1.3, 3.1.4 and 3.1.5 we discuss work on candidate expansion extraction that use machine learning based approaches; In Section 3.1.6, we present the rules used to build a dataset for acronym disambiguation; in Section 3.1.7 we present a work that explores acronym and in-expansion extraction in text written in Hebrew; in Section 3.1.8 and in Section 3.1.9 we describe the most recent works in acronym and in-expansion extraction. Finally, in Section 3.1.10, we summarize and highlight the open problems of acronym and in-expansion extraction.

3.1.1 A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text

Schwartz and Heart [97] focuses on correctly extracting, sometimes called ”identifying”, acronyms (or abbreviations) and their expansions (definitions). The authors discuss an algorithm for biomedical text, but is generalizable. The idea is to use a simple pattern matching algorithm that does not require any training data. Due to its simplicity and its applicability to other domains, this work has been highly cited and is often used as a baseline.

This work follows a three-step process as described in [75]:

1. *Acronym extraction:* find acronyms in a document;
2. *Candidate expansion extraction:* build candidate pairs of acronyms and possible expansions <acronym, expansion> from information in the document alone;
3. *Candidate refinement:* evaluate each candidate pair using a variety of heuristics to obtain a final expansion for each acronym that has at least one candidate expansion within the document.

The first step, acronym extraction, defines the conditions that an acronym has to obey in order to be identified. The authors posit that an acronym (or abbreviation) has a length between 2 to 10 characters, contains at least one letter, the first character must be alphanumeric and contains a maximum of two words.

In the second step, <acronym, expansion> candidate pairs are built. Similar to the approach in [75], this work identifies two forms of acronym-candidate pairs based on two parentheses patterns: either the expansion immediately precedes the acronym between parenthesis, e.g., Heat shock transcription factor (HSF); or the opposite, the acronym immediately precedes its expansion between parenthesis, e.g., HSF (Heat shock transcription factor).

In addition to this parenthetic relationship, the candidate expansion must start with the acronym's first character and be in the same sentence. Further, relative to the number of characters in the acronym, the number of words of the expansion cannot exceed 5 or be more than double that number. Finally, the expansion cannot start and end with prepositions, be-verbs, modal verbs, conjunctions or pronouns.

The next step, candidate refinement, is to find the right subset of words for an expansion given the <acronym, expansion> candidate pair. The authors propose to use a simple pattern matching algorithm that does not require any training data and whose objective is to find the shortest expansion that matches the acronym. The algorithm uses two indices, one for the acronym and another one for the expansion, each index starts in the last character of the acronym or the expansion. It decrements both indices if the character matches the first letter of the word of the candidate expansion, otherwise the algorithm decrements only the expansion index. The algorithm stops when the expansion index reaches the first word (position 0) or when the algorithm matches all characters in the acronym. Additionally, the first character in the acronym has to match the first character of the first word in the expansion. The last word in the candidate expansion has to be in the final expansion. Hyphens (or any other non alphanumeric character) are considered a word boundary when matching the first acronym character, however hyphenated words are extracted together, e.g., 3-N-**m**aleimidyl-proponiony-biocytin (**M**PB).

3.1.2 An Improved Method for Extracting Acronym-definition Pairs from Biomedical Literature

Mohammmed and Nazeer [96] proposes an improvement over Schwartz and Hearst [97] as presented above in Section 3.1.1. For simplicity, we only list and describe the changes proposed to each step of the previous work.

The first step, that finds acronyms in the text, is the same as in previous work. In the second step, where <acronym, expansion> candidate pairs are built, this work allows acronym-candidates that belong to different sentences in the text to form a candidate pair. Additionally, it identifies candidate pairs not only by use of parenthesis patterns in [97] but also by patterns where acronyms and expansions are separated by a hyphen or a colon.

Regarding the third and last steps, candidate refinement, where the algorithm finds the right subset of words for an expansion given the candidate pair, this work proposes the following changes: acronyms and expansions may start with a digit, and if an acronym starts with a digit (e.g., 1), it looks for a valid representation of that digit (e.g., one) in the expansion, e.g., Fifth Generation Language (5GL).

3.1.3 High-recall Extraction of Acronym-definition Pairs with Relevance Feedback

Yarygina and Vassilieva [117] propose another acronym and in-expansion extraction approach. However, their objective is to increase recall, i.e., the number of correctly extracted acronym expansion pairs divided by the number of existing acronym expansion pairs. For that purpose, they built a system whose steps are similar to the other works such as Schwartz and Hearst's [97] and Mohammmed and Nazeer's [96] presented in Sections 3.1.1 and 3.1.2. The main difference consists in candidate refinement step, through the use of machine learning techniques (instead of handmade patterns/rules) and user feedback to increase the training data.

The first step, acronym extraction, proposed additional modifications to the previous work in Section 3.1.1 to increase the number of extracted acronyms. Acronyms are allowed to contain other symbols in addition to parentheses e.g., '.', '-', '=', ',', and '.' and have groups of uppercase letters or digits separated by other symbols e.g., PS Act (Packers and Stockyards Act of 1921) and (R/EOC) Regional/ Emergency Operations Center.

In the candidate expansion extraction step, the authors also proposed changes to previous work to increase the number of candidate pairs (again with the purpose of increasing the recall). In contrast to Schwartz and Hearst [97], which implements an algorithm that uses indices to match acronym characters to expansion characters and where at least one letter in the acronym must be the first letter of a word in the expansion, the authors allow up to 2 missing letters from the acronym in the expansion and require that the Levenshtein distance between the extracted acronym and an acronym generated directly from the expansion must be below 2.

Regarding the last step, the candidate refinement step, the authors consider the candidate selection problem to be a machine learning classification problem, i.e., a pair is true or false. This is the main contribution of this work. The classification is performed using decision trees (the C4.5 algorithm [87]) that receives as input 100 features in total. Those features are: (i) number of symbols, uppercases and lowercases of acronym and expansions, (ii) number of words in the expansion, (iii) number of existing different symbols between acronym and expansion, and (iv) statistics on part of speech in the acronym

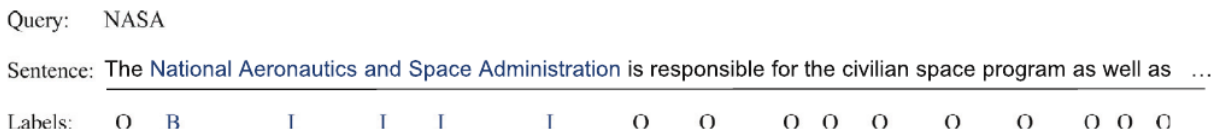


Figure 3.1: Example of identifying the expansion for NASA through a labeling sequence problem as presented in [63]

and in the expansion obtained though OpenNLP[1].

In order to increase performance, this work makes use of user feedback to identify the correct acronym and expansion pairs in documents to retrain the machine learning model. Before presenting a document to the user, the candidate acronym expansion pairs of the document are evaluated by the previously trained machine learning model, here called the global model. The user can give feedback over a part of the document by assigning true or false to candidate pairs. Using the candidate pairs analyzed by the user and the remaining candidate pairs classified by the global model for that document, another machine learning model is trained called a local model. The local model is trained only on candidate pairs from the document. The authors' intention is that it specializes to analyze the correct candidate pairs found in that particular document. Intuitively the patterns used by the writer to specify acronyms and expansions in the document should be consistent. The local model is then used to reclassify the unseen parts of the document. Until the user reaches the end of the document, the system enters in a loop where the user provides feedback over the data, then the local model is retrained with new user feedback and classifies unseen candidate pairs by the user. Finally, after the user finishes classifying the candidate pairs in the document, the global model is retrained using the available training data which includes the new user feedback. Then the user can move to another document and the same process applies.

3.1.4 Multi-granularity Sequence Labeling Model for Acronym Expansion Identification

Liu et al. [63] addressed the expansion extraction (which they call "identification") problem though a machine learning technique called Latent-state Neural Conditional Random Fields (LNCRF) [71]. The authors used an annotated corpus based on Wikipedia[2].

In this work, the problem of expansion extraction is modeled as a sequence labeling problem in order to be able to apply LNCRFs. The sequence (i.e. the text possible containing an expansion) is composed of tokens, and each token is labeled with: **B** for beginning of an expansion, **I** for inside of an expansion, or **O** for other tokens, i.e, that are not part of the expansion. See Figure 3.1 for an example of labeling tokens for finding the expansion of the NASA acronym.

LNCRFs are a variant of the Neural Conditional Random Fields (NCRFs) [71] which are Conditional Random Fields (CRFs) with a neural network layer [57]. In Figure 3.2, we show the architectures of a CRF in (a), a NCRF in (b), and LNCRF in (c), where the variables X identify a set of features for each

[1] https://opennlp.apache.org/
[2] https://www.wikipedia.org/

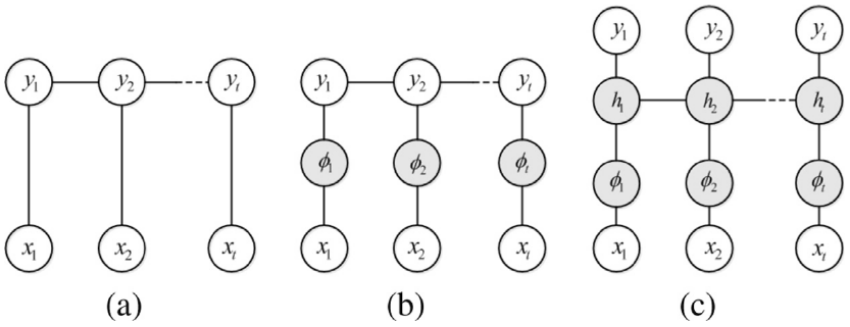


Figure 3.2: CRF (a), NCRF (b) and LNCRF (c) architectures presented in [63]

token, and the variables Y the corresponding labels. As we can see, the set of labels Y in CRFs depend on the neighbors in order to model the sequence dependencies, e.g., if the previous token has a high probability of being labeled with a **B** then it may imply that the next token has a higher probability of being labeled with an **I**.

Regarding the features, the authors used a mixture of three types of features: (i) orthographical features to characterize token information (e.g., capitalized token), (ii) token-query features represent the accordance of a token and the given acronym (e.g., if the first letter in the token exists in the acronym), and (iii) context features that provide information from the neighbor tokens (e.g., if the first letter of the next token appears next in the acronym).

3.1.5 Language Independent Acquisition of Abbreviations

Glass et al. [41] proposed a new machine learning based approach to efficiently extract acronyms/abbreviations and the corresponding expansions from text. With this approach, the authors aimed at building a 7 languages dictionary of acronym and expansions from Wikipedia.

The first part of this work explores the creation of a Wikipedia dataset of acronyms and expansions. The authors obtained a ground truth set from Wikipedia redirection and disambiguation pages from the most used 7 languages in Wikipedia including English, German and non Latin languages such as Russian and Japanese.

Then, the authors propose a *candidate generation* step to obtain acronym and expansion candidate pairs and then a *score* step where candidate pairs containing different scores are evaluated as true or false pairs. The candidate generation step output is the union of the candidates from two distinct techniques. The first one is the simple algorithm of Schwartz and Hearst [97] as presented in Section 3.1.1 augmented to provide a score based on the occurrence frequency of a pair. The second technique is called Candidate System 2 and is an effort to obtain different candidate pairs from the simple algorithm by, for instance. considering other patterns instead of acronym or expansion inside parentheses. Candidate System 2, first scores a pair based on three factors: (i) co-occurrence frequency of pairs in the whole corpus, (ii) minimum distance between occurrences, and (iii) approximate string similarity between acronym and expansion. Then, a threshold is used to accept or reject a candidate pair based on the score.

The *score* step uses different types of scores: (i) scores from the candidate step, two semantic similarities: (ii) the synonym similarity and (iii) topic similarity, and finally (iv) the surface similarity. Those scores are combined by turning them as features to a simple logistic layer (i.e., logistic regression).

The two semantic similarities (i.e., the synonym and topic similarities) are based on word embeddings. The difference between them resides in the source of the word embeddings: the synonym similarity uses the embeddings obtained using Word2Vec, while the topic similarity uses the embeddings obtained using Latent Semantic Analysis (LSA). For both similarities, the score (similarity) is given by the cosine similarity between the acronym embedding and the average of the token embeddings in the expansion.

The surface similarity is the maximum score over all possible alignments between an acronym and an expansion. A possible alignment is a mapping of acronym characters to expansion characters. The authors obtained for each possible alignment, the following features: (i) number and percentage of acronym characters that match the first token expansion character, (ii) percentages of expansions and acronym tokens with no matched characters, and (iii) number of swaps from the matches between acronym and expansion characters. Using those features, a linear model predicts the similarity score for each possible alignment. The final surface similarity score is the highest score predicted.

3.1.6 Acronym Disambiguation: A Domain Independent Approach

In order to test their out-expansion (acronym disambiguation) approach, the authors of [107] proposed an approach to extract expansions from Wikipedia articles. The acronyms are obtained from the Wikipedia disambiguation pages.

To extract an expansion for an acronym, the authors used the following rules: (i) the characters in the acronym have to match in sequence the first characters of the expansion words; (ii) word boundaries are spaces, underscore or dash; and (iii) a word in the expansion whose first character does not match must be a stop word.

3.1.7 Acronyms: Identification, Expansion and Disambiguation

The work present at [50] focused on techniques for acronym and in-expansion extraction (which they call ”identification”) for texts written in Hebrew. The novelty of this work is on building a dictionary of acronyms and expansion namely a database containing the result of the extraction of valid acronyms and corresponding expansions in text. Most of the techniques proposed in this work are designed from scratch and so take advantage of the Hebrew language structure. For instance, in Hebrew the language grammar is very clear regarding the use and generation of acronyms (i.e, have a particular format) but not so straightforward in the opposite way where given an acronym we want to find the corresponding expansion because the expansion may include word prefixes and suffixes (which is how the language defines connection works such as *of, for,* or *a*nd). Still, the authors were able to define formation rules from expansions to acronyms.

To build the acronym expansion dictionary, first, the system finds the acronyms (easy due to the

language structure) and then, in a kind of reverse engineering, it uses the formation rules to obtain candidate expansions from the whole corpus texts. These candidate expansions are then filtered out by a previously trained binary Support Vector Machine (SVM) over a human annotated corpus (also used as a gold standard). The SVM uses a set of features from acronyms and expansions lengths, word and documents frequencies, and also Latent Dirichlet Allocation comparison between the candidate expansion and the acronym. The authors also limited their system to just expansions and acronyms that occur at least 5 times in the corpus.

3.1.8 SciDr at SDU-2020: IDEAS - Identifying and Disambiguating Everyday Acronyms for Scientific Domain

In SciDr, Singh and Kumar [99] interpret acronym and in-expansion extraction as a sequence labeling problem for their SciDr in-expander (SciDr-in). Liu et al. [63], presented in Section 3.1.4 have previously address the in-expansion extraction as a sequence labeling problem, but not the acronym extraction. Singh and Kumar [99] sequence labeling consisting of three tags identifying: (i) a word in an acronym, (ii) a word in an expansion and (iii) other word.

Due to their participation in the SDU@AAAI competition [111] for in-expansion extraction, the authors tested different models based on pre-trained language model (e.g., BERT [28] or SciBERT [9]) together with Conditional Random Fields (CRFs) [57] to classify each word with one of the three tokens.

The best solution using a single model was obtained with SciBERT [9] which is a language model based on Transformers and pre-trained on research papers from Semantic Scholar [3]. The SciBERT used was fine-tuned with training data for the sequence labeling task.

To obtain even better performance, Singh and Kumar [99] implemented in SciDr-in a blending process [98]. It splits the training data into train and validation sets. Five different SciBERT models (e.g., number of epochs and learning rate values) are constructed based on the training set. Predictions on the validation set are then stored. The models, the predictions, and additional syntactic features extracted from the word-to-tag mapping are used to train 5 Conditional Random Fields (CRFs) [57] in a 5-way cross-validation setting. The resulting CRF models are ensembled using hard voting.

3.1.9 MadDog: A Web-based System for Acronym Identification and Disambiguation

Veyseh et al. [110] created for MadDog system an in-expander that introduces minimal variations of the Schwartz and Hearst technique [97] (described in Section 3.1.1). This in-expander follows the same exact three steps followed by Schwartz and Hearst [97]. For simplicity, we only list the differences proposed to each step.

For the first step, the Acronym extraction that finds acronyms in text, MadDog [110] adds the restriction that 60% of the acronym's characters must be upper-case letters.

[3] https://www.semanticscholar.org/

The second step, the builds the candidate pairs of acronym and expansions was not modified. However, Veyseh et al. [110] added priority rules on Schwartz and Hearst [97] technique third step, the candidate refinement, in order to improve precision. If a rule is fulfiled then it returns the expansion, otherwise continues until an expansion is found or discarded. The rules are the following:

1. The upper case first letter of each word in the candidate definition match the acronym characters
2. The first letter (perhaps lower case) of each word in the candidate definition match the acronym characters
3. Schwartz and Hearst [97] technique modified to discard the last words of the candidate expansion if no character matches an acronym character

3.1.10 Discussion

In this section, we covered the related work concerning acronym and in-expansion extraction. Table 3.1 presents a summarized comparison between those works. The various works present a similar three-step process: (i) Acronym Extraction where are acronyms are extracted, (ii) Candidate Expansion Extraction where expansion candidate pairs are generated, and (iii) Candidate Refinement where the final pairs are selected. The exceptions are Liu et al. [63] and Thakker et al. [107] that perform expansion extraction and refinement (second and third steps) in a single step; Glass et al. [41] and Singh and Kumar [99] that performs those steps using the same techniques. Additionally, we list the datasets used in the experimental evaluations of related work.

First and second steps are essentially rule/pattern based approaches. Some were more specific using only parenthesis patterns as in Schwartz and Hearst [97] (Section 3.1.1). Regarding the third step, Schwartz and Hearst [97], Mohammmed and Nazeer [96], Thakker et al. [107], and Veyseh et al. [110] (Sections 3.1.1, 3.1.2, 3.1.6, and 3.1.9) proposed hand-made algorithms, whether in more recent works Yarygina and Vassilieva [117], Glass et al. [41], and Jacobs et al. [50] (Sections 3.1.3, 3.1.5, and 3.1.7), the authors consider the candidate selection problem to be a classification problem, i.e., a pair is true or false. The exception is Liu et al. [63] and Singh and Kumar [99] (Section 3.1.4 and 3.1.8) which do not follow the three-step approach, instead they model the expansion extraction as a sequence problem where machine learning is used to classify tokens as if they belong to the expansion of a particular acronym or not. Additionally, Singh and Kumar [99] also extracts acronym using the same token classification approach.

Most of these works have used Schwartz and Hearst [97] as a baseline, Mohammmed and Nazeer [96] and Veyseh et al. [110] proposed some improvements to the Schwartz and Hearst's approach. Mohammmed and Nazeer [96] compares their proposal against it using the same dataset about biomedical abstracts.Veyseh et al. [110] compares Schwartz and Hearst [97] and Liu et al. [63] in the SciAI dataset from [84]. In Yarygina and Vassilieva [117] changes were also proposed in the two first steps in order to improve recall and machine learning on the third step that takes advantage of user feedback tested in

Table 3.1: Summary of related work in acronym and in-expansion extraction.

Related Work	Acronym Extraction	Candidate Expansion Extraction	Candidate Refinement	Datasets
Schwartz and Hearst [97]	- Length between 2 to 10 characters - Contains at least a letter - The first character must be alphanumeric - Contains a maximum of two words	- Parenthesis patterns - Acronym and expansions in the same sentence	- Shortest expansion that matches the acronym - Expansion must start with the acronym's first character	MEDLINE
Mohammmed and Nazeer [96]	- Schwartz and Hearst [97]	- Parenthesis, hyphen, and a colon patterns - Acronym and expansions may be in different sentences	- Schwartz and Hearst [97] + can start with a digit + digit matches with any digit representation (e.g, III)	MEDLINE
Yarygina and Vassilieva [117]	- Schwartz and Hearst [97] + can contain other symbols instead of '()'	- Schwartz and Hearst [97] + match can have to 2 missing letter + Levenshtein distance must be below 2	- Decision trees classifier with string metadata and statistics on part of speech as features - user feedback	Medstract, military and disa
Liu et al. [63]	N/A	- LNCRF sequence labeling with orthographical, token-query, and context features		Wikipedia based dataset
Glass et al. [41]	- Union of: - Schwartz and Hearst [97] - Candidate system 2 which is based on tokens co-occurrence		- Logisitc regression classifier with candidate step, synonym, topics and surface scores as features	Wikipedia articles in 7 languages
Thakker et al. [107]	N/A	- The first characters in each expansion token must match the acronym characters - Except if the expansion token is a stop word - Word boundaries are spaces, underscore or dash		N/A
Jacobs et al. [50]	- Hebrew grammar	- Reverse engineered acronym formation rules	- SVM classifier with acronym and expansion lenghts, frequencies, and LDA as features	Hebrew dataset
Singh and Kumar [99] (SciDr-in)	- Blending of SciBERT models followed by CRFs for a sequence labeling of both acronyms and expansions			SciAI version of [111]
Veyseh et al. [110] (MadDog-in)	- Schwartz and Hearst [97] + 60% of the acronym's characters must be upper-case	- Schwartz and Hearst [97]	- priority rules that favor expansion words whose first character matches the acronym - Then Schwartz and Hearst [97]	SciAI version of [84]

biomedical and defense report datasets. Glass et al. [41] proposes a new dataset based on Wikipedia and achieves better performance than Schwartz and Hearst [97] using a scoring system. Glass et al. [41] tried to replicate the features used in Liu et al. [63] but with no success. This last work report experiments over an annotated and publicly unavailable corpus based on Wikipedia against other machine learning techniques such as Support Vector Machines (SVMs) and their previous work on Conditional Random Fields. Thakker et al. [107] did not perform extraction experiments and they used the proposed expansion extraction rules to build an out-expansion dataset. Jacobs et al. [50] focused on Hebrew language and so did not compare with related work used for English. Singh and Kumar [99] was only evaluated in the context of the SDU@AAAI competition [111].

It is clear by now that there is no best approach for acronym and in-expansion extraction. When evaluated together, the experiments only included a limited number of datasets from specific domains. Even the works of Singh and Kumar [99] and Veyseh et al. [110] that used the SciAI dataset, their experiments are in different versions of the mention dataset. Moreover, in practice there is no evidence that the overhead of using training data and the execution time spent in machine learning based techniques overcomes rule-based approaches in this task.

3.2 Out-expansion

In this section, we review work related to out-expansion. This work uses the subject matter of a document to identify expansions for ambiguous acronyms, i.e., acronyms that are not defined in the document and for which we know several expansions. In Sections 3.2.1 and 3.2.2, we present two works that explore word or document embeddings for out-expansion; in Section 3.2.3 we present a work that explores out-expansion in text written in Hebrew; and in Section 3.2.4 we detail a work that uses word embeddings to out-expand acronyms in captions.In Section 3.2.5 and in Section 3.2.6, we describe the most recent works in acronym and in-expansion extraction that use neural network models. Finally, in Section 3.4.7, we summarize and highlight the open problems of out-expansion.

3.2.1 Acronym Disambiguation Using Word Embeddings

Li et al. [61] proposed two approaches based on word embeddings from Word2Vec to address the out-expansion (acronym disambiguation) problem. These approaches combine word embeddings (i.e., word vectors) from the acronym or expansion context to generate an acronym or expansion embedding. Intuitively, suppose that the same acronym can be expanded in different ways depending on the context, i.e., an ambiguous acronym. Suppose further that the expansion extraction also keeps track of the context of each expansion by generating an expansion embedding. Then, the out-expander will compare the context of the acronym with all the contexts of the expansions and choose the expansion having the closest context. In this work, contexts are compared by computing the cosine similarity between an acronym embedding and an expansion embedding.

The first approach to compute the embeddings for a term (an acronym or an expansion) called TF-IDF Based Embedding (TBE) is based on TF-IDF top words (Section 2.1.2). For each top n TF-IDF word of each document containing a term t (acronym or expansion), its embedding (word vector from Word2Vec) is multiplied by the corresponding TF-IDF weight. All these weighted embeddings are summed together to obtain the final term t embedding.

Regarding the second approach, called Surrounding Based Embedding (SBE) instead of using TF-IDF, it combines the embeddings of the words surrounding the term. To do so, it first obtains the embeddings of all the words found inside of a window w around any expansion occurrence in all documents or the words around any occurrence of an acronym within the same document. Then, it sums the embeddings together to obtain the final expansion or acronym embedding.

3.2.2 Acronym Disambiguation: A Domain Independent Approach

Thakker et al. [107] proposed another approach for out-expansion (acronym disambiguation) based on embeddings. In concrete, they explore Doc2Vec to create embeddings for each document, i.e., document vectors. They also proposed an approach to extract acronyms and expansions from Wikipedia articles described in Section 3.1.6.

Their out-expansion approach obtains, for each occurrence of a term (acronym or expansion), a window of characters around them (the authors used 2000 to 5000 characters) here called a context embedding. For each acronym and corresponding set of candidate expansions, the system trains a Doc2Vec model using that acronym's context. Then, when disambiguating an acronym, the system compares the context embedding of the acronym against the context embedding of its candidate expansions using cosine similarity. It then selects the expansion whose context embedding is the most similar to the acronym context embedding. Note that this work uses a Doc2Vec model per acronym and not a single general model for the whole corpus in contrast to Word2Vec models used in SBE [61] and Charbonnier et al. [17] (Sections 3.2.1 and 3.2.4).

3.2.3 Acronyms: Identification, Expansion and Disambiguation

Apart from the techniques for acronym and in-expansion extraction described in Section 3.1.7, Jacobs et al. [50] also focused on out-expansion (acronym disambiguation) techniques for texts written in Hebrew.

Regarding the out-expansion of ambiguous acronyms in Hebrew (a not so common case in this language because most acronyms found have just one expansion), the authors ranked the expressions by their source: first, if they are of type gematria (a type of acronyms for numeric expressions such as calendar years), second, if they come from the annotated corpus (gold standard), and then finally if they were identified by the Support Vector Machine (SVM) model as described in Section 3.1.7. Expansions coming from the same source are ordered by an LDA-based score. This score is obtained using the cosine similarity to compare the combination of LDA topics in each word of the expansion against the document topic vector containing the ambiguous acronym.

3.2.4 Using Word Embeddings for Unsupervised Acronym Disambiguation

Charbonnier et al. [17] proposed an out-expansion (acronym disambiguation) approach based on word embeddings weighted by TF-IDF scores to disambiguate acronyms in scientific article captions. Expanding acronyms in captions is important for these authors because this work was part of project NOA (Reuse of Open Access Images)[4] that provides a web page to search for images based on terms. In the context of project NOA, assigning an expansion to an acronym is important to obtain more meaningful information from the available terms from captions.

Regarding the out-expansion approach, the authors consider the word embeddings from Word2Vec and TF-IDF for the whole corpus. Alternatively, they also tried word embeddings from pre-trained Word2Vec model but found that the pre-trained models did not provide better results. To obtain an embedding for an acronym in a caption, the remaining tokens in the same caption are combined.

The computation is the sum of all token embeddings multiplied by the inverse document frequency value (IDF part of TF-IDF). The expansion whose embedding is the most similar with the acronym embedding in terms of cosine similarity is selected.

3.2.5 Identifying and Disambiguating Everyday Acronyms for Scientific Domain

SciDr purposed by Singh and Kumar [99] also provides an approach for the out-expansion task of the SDU@AAAI competiton [111]. The authors use SciBERT models fine-tuned with different hyperparameters assembled together for the competition.

The authors, formulate the out-expansion problem as a substring prediction task where the objective is to find the substring that matches the best expansion for a given sentence. The input is a list of expansions concatenated with a Separating Token, $[SEP]$, followed by a sentence as input, for example for acronym *PDF* an input would be *portable document format probability density function* $[SEP]$ *formats including the PDF*. Then, it uses a language model (e.g., BERT [28] or SciBERT [9]) whose output is an encoding that is passed to a dense linear neural network layer that provides two outputs: the start and end indexes corresponding to the substring from the original input. It may be the case that the substring doesn't perfectly match an expansion (e.g., document format probability), in this situation, it computes the Jaccard similarity of each expansion against the substring, and the most similar expansion to the substring is chosen.

The authors tried different models: (i) BERT, (ii) SciBERT, and (iii) SciBERT with external data constituted by Wikipedia pages that contain an expansion found in the SDU@AAAI training data. They also explored an ensemble approach of models (ii) and (iii), the ensembling was performed by 5-cross validation for each (ii) and (iii) resulting in 10 models. Then, to obtain the final output, the average of the probabilities for the substring indexes predicted by each model is taken before computing the Jaccard similarity.

Finally, the authors used a post-processing step for the competition which search in the input sentence if an expansion for an acronym is present, if found selects that expansion as the output. Otherwise,

[4] https://noa.wp.hs-hannover.de/

runs the ensemble process mentioned above.

3.2.6 MadDog: A Web-based System for Acronym Identification and Disambiguation

In this section, we describe the out-expander technique included in the MadDog system [110] that also processes a sentence as input. The authors processed a large number of texts from English Wikipedia[5], Arxiv[6], Reddit[7] and two sources of textual documents from biomedical data (Medline abstracts[8] and the PMC open-access subset[9]). To extract the acronyms and expansions and build a training dataset for out-expansion, the authors used their in-expansion technique described in Section 3.1.9.

Several neural network models are used to classify a sentence with an expansion. Each model is trained with 100k of sentences (samples). A sentence is tokenized and the expansion occurrences replaced by the acronym. First, each token is replaced by its GloVe embedding [78] and then they are placed into a sequential model (a Bi-directional Long Short Term Memory network) that produces the embeddings for the sentence and the acronym. Finally, those embeddings are used as features for a feedforward network to classify the input sentence with an expansion.

3.2.7 Discussion

In this section, we covered the related work concerning out-expansion. Table 3.2 presents a summarized comparison between the out-expansions approaches proposed by those works and lists the datasets used on their experimental evaluations.

Li et al. [61] (Section 3.2.1) compared their out-expansion approaches against the Classic Context Vectors and previous related work using two annotated datasets based on articles abstracts: MSH contains biomedical articles, and ScienceWISE physics article abstracts. They report that their Surrounding

[5] https://en.wikipedia.org/
[6] https://arxiv.org/
[7] https://www.reddit.com/
[8] https://www.nlm.nih.gov/medline/medline_overview.html
[9] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

Related Work	Out-expansion Approach	Datasets
Li et al. [61] (SBE)	Cosine similarity of combinations of Word2Vec embeddings	MSH and ScienceWISE
Thakker et al. [107]	Cosine similarity of local Doc2Vec embeddings	Computer Science acronyms from Wikipedia
Jacobs et al. [50]	Source priority and then LDA-based score	Hebrew dataset
Charbonnier et al. [17]	Cosine similarity of combinations of Word2Vec and TF-IDF embeddings	NOA (acronyms in captions)
Singh and Kumar [99] (SciDr)	Ensembler of SciBERT models that output a substring with the expansion	SciAD version of [111]
Veyseh et al. [110] (MadDog)	Sequence model with a feedforward network classifier	SciAD version of [84]

Table 3.2: Summary of related work in out-expansion.

Based Embedding (SBE) approach outperforms related work but does not obtain better results than the Classic Context Vectors on the MSH dataset.

Thakker et al. [107] (Section 3.2.2) uses a dataset composed of acronyms used in the computer science field, but also expansions from other fields. This dataset was built based on the authors' extraction rules (Section 3.1.6) applied to Wikipedia articles. It uses a Doc2Vec model per acronym and not a single general model for the whole corpus as is common to produce accurate semantic embeddings (e.g., Word2Vec models used in SBE [61] and Charbonnier et al. [17]). Further, it may allow different expansions within a single document for the same acronym, which is contrary to the intuition that each acronym will normally have only one expansion within a document.

Jacobs et al. [50] and Charbonnier et al. [17] (Sections 3.2.3 and 3.2.4) present two recent works with interesting and novel approaches, however they are applied to very specific domains and problems: Jacobs et al. [50] explore acronyms in Hebrew texts and Charbonnier et al. [17] expands acronyms in captions.

Recently, Singh and Kumar [99] and Veyseh et al. [111] (Sections 3.2.5 and 3.2.6) proposed two out-expansion approaches based on neural networks. They were both tested for different versions of the SciAD dataset. Moreover, the approach of Veyseh et al. [110] used only their large dataset as training data, instead of the SciAD training set.

The above work on out-expansion has many clever ideas, but there is no study that compared the works with one another on plain English text documents. Comparison among document representation techniques should also take into account a common preprocessing which is not done in Li et al. [61].

Out-expansion works did not explore the direct (or standalone) use of the document representation techniques TF-IDF and LDA for English. Li et al. [61] and Charbonnier et al. [17] tried combinations of Word2Vec with TF-IDF. Jacobs et al. [50] included LDA as part of their disambiguation solution for Hebrew texts. Thakker et al. [107] used local Doc2Vec models as a context. However, no global Doc2Vec model was tested nor was the use of the whole document as a context.

Further, regarding the comparison of embeddings, the only similarity approach used was cosine similarity. None of these works tried a machine learning approach for this purpose, despite the fact that selecting an expansion could be seen as a classification problem, i.e., labeling acronyms with an expansion.

3.3 End-to-end Acronym Expanders

To our knowledge, systems that expand abbreviations and/or acronyms use a pre-defined dictionary of acronym-expansions [1, 42] as opposed to trying to discover the proper expansion based on context.

Only two end-to-end systems use context for out-expansion. First, Ciosici and Assent [20] propose an end-to-end abbreviation/acronym expansion system architecture that performs out-expansion. Unfortunately, their demo paper does not provide enough technical details and their code is proprietary.

Most recently, Veyseh et al. [110] proposes an end-to-end acronym expansion system, called MadDog, which contains a rule-based in-expander technique that improves on [97] and an out-expander

based on neural networks: a sequential model to encode context followed by a feedforward network to classify the input with an expansion. They also trained their models on a large corpus of sentences.

None of the these two systems provides a framework with easy plug-in for different in and out-expansions techniques nor uses other data sources. Moreover, none of them were evaluated on an end-to-end acronym expander benchmark.

3.4 User Involvement in Data Cleaning

In this section, we focus on the related work that involves the user in a data cleaning process in such a way that she/he can provide feedback to repair the data. In Section 3.4.1, we present a framework based on data quality rules, named LLUNATIC, that takes into account the user feedback to resolve conflicts in data repairs. Furthermore, we also detail two frameworks that use Machine Learning (ML) models trained with user feedback to improve the data cleaning process: Guided Data Repair presented in Section 3.4.2 that relies on the user to choose which data repair to apply; and Continuous Data Cleaning presented in Section 3.4.3 where the user can decide which type of repair to search. In Section 3.4.4, we present the DANCE framework that resolves data inconsistencies using only user feedback. In Section 3.4.5 and in Section 3.4.6, we describe FALCON and ICARUS respectively, which are two data cleaning frameworks that search for a small set of Structured Query Language (SQL) update statements that can clean the data by modifying cell values. Finally, in Section 3.4.7, we summarize and highlight the open problems of these works.

3.4.1 LLUNATIC

LLUNATIC [39] is a data cleaning and mapping framework that models a data cleaning (and/or mapping) program using data rules. This framework generalizes the data cleaning process based on data rules and provides extendable components for: (i) identifying rule violations, (ii) searching for the available data repairs that can resolve the violations, and (iii) selecting data repairs based on some criteria (usually the minimum number of changes applicable to the whole dataset that satisfy the rules).

LLUNATIC introduces a language based on equality generating dependencies (egds) [8] whose semantic incorporates most of the data quality rules from the literature. Moreover, because Llunatic is also a mapping framework, this language also enables the user to specify *tuple generating dependencies* [32] that support data mappings which is out of the scope of this thesis. Egds generalize the data quality rules into a common and unified syntax structure (see more details and an example in Section 2.2.2). This unification enables the LLUNATIC framework to explore performance improvements by providing optimizations over this common language structure. The authors concern was also to add support so that cleaning specifications could accommodate user preferences for data repairs to define the best set of data repairs in alternative terms (instead of just minimum number of repairs), e.g., give preference

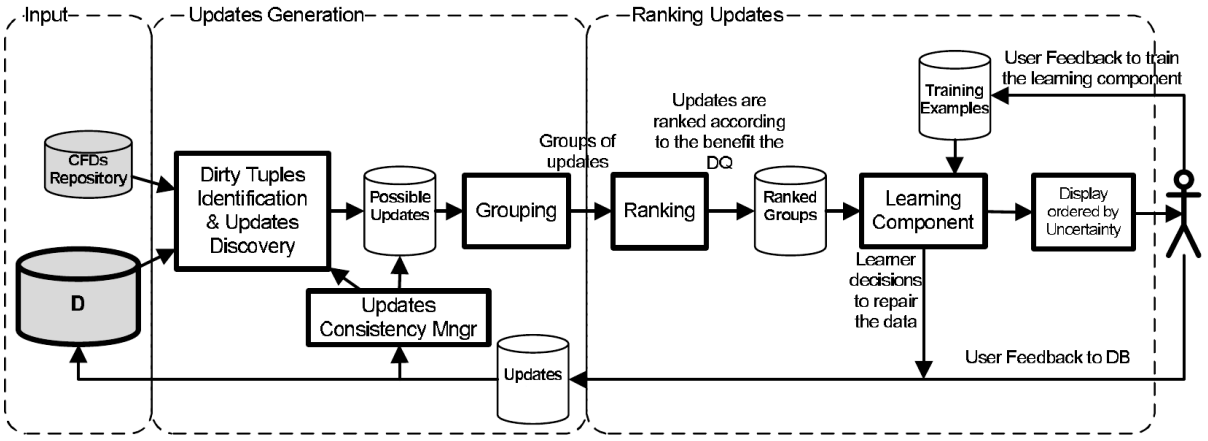


Figure 3.3: GDR framework as presented in [114].

to maintain values from a particular column over another. Therefore, it introduces the notion of partial order that allows the user to prioritize the repairs by specifying, among attributes, which ones contain the preferred values over others. LLUNATIC contains a component called *Cost Manager* that implements an optimization procedure to prune the search for data repairs, thus enables the chase (search for data repairs) to scale with more data. It also allows the user to tune this optimization procedure (based on different heuristics).

The LLUNATIC framework aggregates cells into cell groups which are sets of cells that must have the same value. Cells are grouped according to the data repair that is used to solve the violations of data quality rules, e.g., two publication records with the same journal name and volume number that are not published in the same year are grouped together in order to get both the same value in year. Values can be nulls, constants or lluns. *Lluns* are a special type of values that represents a set of possible values for a cell or cell groups which could not be selected by the cost manager and the partial order (user preferences), so they require user feedback. After LLUNATIC finishes to run, the user can inspect the chase as a tree and provide a value for each llun. This value is then replaced in all cells of the group in the database.

3.4.2 Guided Data Repair

Guided Data Repair (GDR) [114] is a framework that takes advantage of user feedback to improve the selection of data repairs. Data repairs in GDR are generated using the algorithm to find data repairs developed at [23] that generates data repairs for cleaning violations of Conditional Functional Dependencies (CFDs). Instead of using heuristics to decide which data repairs to apply, GDR resorts to the user feedback to find the optimal subset of data repairs to apply to the data. Moreover, while requesting for user feedback, GDR uses this feedback to train Machine Learning (ML) models capable of replacing the user.

Figure 3.3 shows the different components of the GDR framework grouped into three super components: *Input*, *Updates Generation*, and *Ranking Updates*. The objective of GDR is to modify a database D in order to satisfy a set of CFDs Σ stored in a *CFDs Repository*. The set of tuples that are involved

in the violation of at least one CFD are considered dirty. The *Dirty Tuples Identification and Updates Discovery* component uses the techniques presented at [23] to discover the set of tuples considered dirty, and then, to generate data repairs in the form of tuple updates to D identified as the *Possible Updates*. When an update from the *Possible Updates* set is chosen, either through user feedback or through a ML model prediction, the *Updates Consistency Manager* component updates the database D and generates a new set of dirty tuples and, consequently, new *Possible Updates*. Note that, although modifications to D are intended to resolve violations, they can indeed cause the occurrence of further violations of other CFDs. Given the *Possible Updates*, GDR groups together the candidate data repair updates that provide the same attribute value, independently of the target tuple and original values. These groups will allow the user to visualize similar updates together and will also help ML algorithms in finding data patterns.

The novelty of GDR regarding other data cleaning works resides in the set of components grouped inside of the *R*anking Updates super component in Figure 3.3. The *Ranking* component ranks the groups generated by the *Grouping* component (inside of the *Updates Generation* super component) into *Ranked Groups* based on the values of the *Valuable Of Information* measure [94] that quantifies the overall benefit of selecting the true data repairs. For the updates in a group, the *Learning Component* uses active learning to guess the user actions and to provide an uncertainty score. The *Learning Component* uses random forest trees, a ML technique that trains a set of ML tree models then, the most frequent output is used as the predicted value. Random forests are further used by the query by committee strategy in the *Learning Component* to compute an uncertainty score. The *Display ordered by Uncertainty* component uses the uncertainty score to sort the updates and then displays them to the user. Afterwards, the user selects one of the top ranked groups to inspect and then provides feedback. The user feedback is given through the following actions on tuple updates: (i) accept the update, (ii) reject the update or (iii) retain the tuple's original values. The user will give feedback on the most uncertain update action guessed by the ML models. When the user feedback is produced, GDR generates *Training Examples* that are used to re-train the ML model in the *Learning Component*. This model is then used for new predictions and new uncertainty scores. Finally, when the user is satisfied with the ML model performance, she finishes the group inspection by setting the ML model to guess the user actions for the remaining updates and then she moves to the next ranked group of possible tuple updates. The data cleaning process finishes when there are no more groups to be visited by the user.

3.4.3 Continuous Data Cleaning

Continuous Data Cleaning [112] is a data cleaning framework which handles stream data and adapts itself to new data by adjusting data quality constraints. Continuous Data Cleaning can apply two type of repairs: (i) data repairs which modify the data to agree with the data quality constraints and (ii) constraint repairs which modify the constraints to agree with the data. This framework generates a set of repairs and then asks the user to select which repairs to apply. The repairs selected by the user are used to train

tid	First_name	Surname	BranchID	Salary	Zip	City
t_1	James	Brown	107	70K	50210	Miami
t_2	Craig	Rosberg	107	50K	50210	Miami
t_3	James	Brown	308	70K	21100	Atlanta
t_4	Monica	Johnson	308	60K	21100	Houston
t_5	James	Brown	401	80K	65300	NY
t_6	Monica	Johnson	401	100K	65300	NY
t_7	Craig	Rosberg	401	80K	65300	Boston
t_8	Mark	Douglas	401	130K	65300	NY

Figure 3.4: Employees example presented in [112]

a Machine Learning (ML) model that, for new violations, predicts the type of repairs (i.e., data and/or constraint repair) to be applied. The framework takes into account the ML model output (i.e., type of repair predicted: data repair, constraint repair or both) to search only for repairs whose type is given by the ML output. In the case that the ML model output is just one type of repair, then the framework searches only for repairs of that type instead of performing an extensive search by generating data and constraint repairs. Note that the Continuous Data Cleaning framework improves the work presented in [18] that always generates data and constraint repairs. By adding data repairs selected by the user and ML models that predict the repair type, the Continuous Data Cleaning framework improves the data quality of a database and enables a faster execution.

Continuous Data Cleaning focus on Functional Dependencies (FDs) and constraint repairs that turn FDs into more specific FDs, i.e., adds left hand side attributes. As an example, consider the table in Figure 3.4 that contains the salaries of employees belonging to a company with multiple branches and the FD `[First name, Surname] ->[Salary]`. This FD is violated by the following set of tuples {t1, t3, t5}, {t2, t7}, and {t4, t6}. A constraint repair must be applied by modifying the original FD into `[First name, Surname, BranchID] ->[Salary]`. This new FD is satisfied by all tuples of the table.

The input of the ML model is a set of 22 features that describe a violation based on statistics regarding: (i) the FD (e.g., `[First name, Surname, BranchID] ->[Salary]`), (ii) the tuples that violate the FD, and (iii) the tuples that share the same values in the attributes involved (either right or left side) in the FD (e.g., in Figure 3.4 tuples that refer to James Brown: t1, t3, and t5). Those features belong to one of five groups: (i) statistics over the violated FD, (ii) statistics over the tuples that violate the FD, (iii) statistics over the tuples that violate other FDs, (iv) statistics over possible constraint repairs, and (v) statistics over possible data repairs. Most of these statistics are computed using accumulator variables, i.e., do not need to compute every time over the whole dataset, thus avoiding to degrade performance.

The ML model output is one of seven available classes that indicates if a violation: (i) cannot be repaired, (ii) is repaired completely by FD repairs, (iii) is repaired completely by data repairs, (iv) is repaired completely by data and FD repairs, (v) is repaired partially by FD repairs, (vi) is repaired partially by data repairs, or (vii) is repaired partially by data and FD repairs. For ML training purposes, the class of a violation (i.e., ML model output) is obtained by the framework by finding the class whose description (e.g., is repaired completely by FD repairs, is repaired partially by data and FD repairs) is closer to describe the set of repairs (composed of constraint and data repairs) selected by the user to fix a violation. The ML model is iteratively trained using the logistic regression technique.

Figure 3.5 shows the Continuous Data Cleaning framework pipeline. The process starts with the user

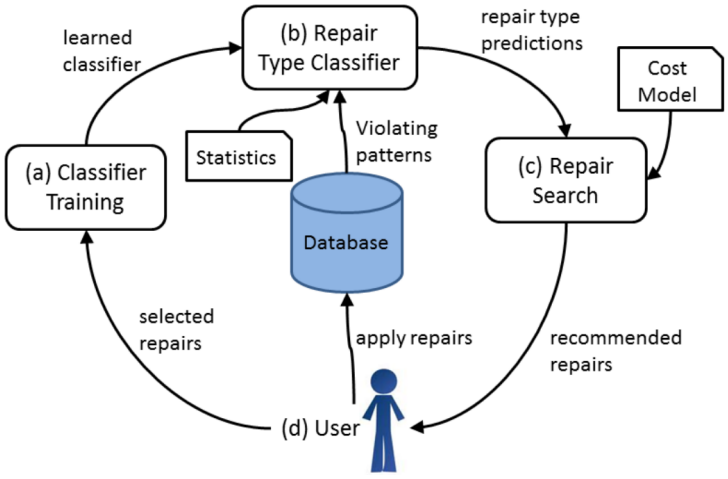


Figure 3.5: Continuous Data Cleaning presented in [112]

selecting the repairs to apply to the data and/or constraints. In step (a), the *Classifier Training* component uses these repairs to train a classification ML model. Then, in step (b), for a given violating pattern, the *Repair Type Classifier* component uses the classification ML model to predict the type of repairs using statistics. Step (c) consists on the *Repair Search* component that searches only for the predicted type of repairs from step (b), i.e., data repairs, constraint repairs, or both. The search is performed by using the method proposed at [18] that uses a cost model to identify the best repairs. Finally, in step (d) the set of repairs generated from the repair search module are shown to the user who selects the subset to apply.

3.4.4 DANCE

DANCE [5] is a data cleaning framework that enables the user to clean data inconsistencies that violate data quality constraints. By guiding the user, this tool minimizes the number of tuples verified by the user to resolve data quality constraints that specify data inconsistencies. In order to reduce the user intervention, DANCE identifies the most probable tuples, whose repair resolves the data inconsistencies, to present to the user. Moreover, DANCE can also clean data in a Query Oriented Data Cleaning System [10] scenario, where an expert identifies incorrect data that resulted from the execution of a query over a database in some context. DANCE converts the identified incorrect data to data quality constraints and executes a data cleaning process where the user is guided to resolve those constraints (i.e., incorrect data).

In Figure 3.6 we present a sample of the UEFA Champions League database that contains football teams in table Teams and matches played for the UEFA Champions League in table Games. This database has two constraints represented by the following rules:

Rule #1: teams from the same country cannot play with each other in Group Stage;

Rule #2: if a country is listed in the Countries table then it must have at least a team from this country in the Teams table.

Games

team1	team2	t1_goals	t2_goals	stage
Celtic	Manchester City	3	3	Group Stage
Celtic	Hapoel Beer Sheva	5	2	Qualification

Teams

name	country
Celtic	UK
Manchester City	UK
Hapoel Beer Sheva	Israel
CSKA Moscow	Russia

Countries

name	num_of_teams
Israel	0
UK	5
England	4
Scotland	1

Figure 3.6: Example UEFA Champions League presented in [4]

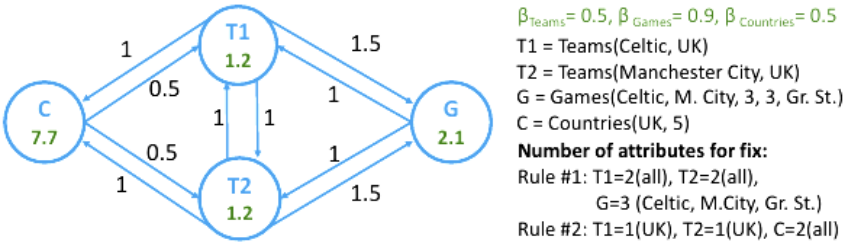


Figure 3.7: Suspicious tuples graph example for UEFA Champions League example presented in [4]

In the Games table, notice that Celtic played with Manchester City in a Group Stage but in the Teams table Celtic and Manchester city are teams from the same country, UK. These data tuples clearly violate Rule #1. DANCE extends the notion of violating tuples (see Section 2.2.2) into *suspicious tuples* which are the tuples that, indirectly, due to a constraint, force a violating tuple to have a specific value, otherwise this constraint is violated. Examples of suspicious tuples are the tuples identified with a darker background color in Figure 3.6. Note that tuple UK in table Countries is not directly involved in the violation of Rule #1 but due to Rule #2 it forces the existence of teams from UK in the Teams table.

In order to decide which suspicious tuples to present first to the user, DANCE creates a graph where nodes are tuples and edges represent the likelihood of a modification (update or delete) of a tuple that leads to the resolution of a violation involving another tuple. Edges are also weighted by the tuple source table reliability β, e.g., in the UEFA Champions League example we are confident that data in Games table is more accurate (i.e., cleaner) than the remaining tables. Then, DANCE runs a PageRank like algorithm [16] that ranks nodes by assigning weights to the nodes based on the edge values. In Figure 3.7, we present a graph for the suspicious tuples in the UEFA Champions League example after running the PageRank like algorithm, the tuples (i.e., graph nodes) and violations are detailed on the right. The first node in the rank, i.e., the node with the highest weight corresponds to the tuple whose modification can fix the most number of violations (e.g., node C = Countries(UK,5) in the graph example). In Figure 3.7, the first tuple to be verified by the user is the tuple from the Countries table with value UK (i.e., node C) that is in fact a tuple that must be removed because the Celtic and Manchester City teams in the context of this UEFA competition belong to distinct federations/Countries (i.e., Scotland and England).

In Figure 3.8, we present the DANCE framework architecture. The main flow of this system (continuous lines) initializes with a database *DB* to clean and a set of integrity *Constraints*. First, the *Violations*

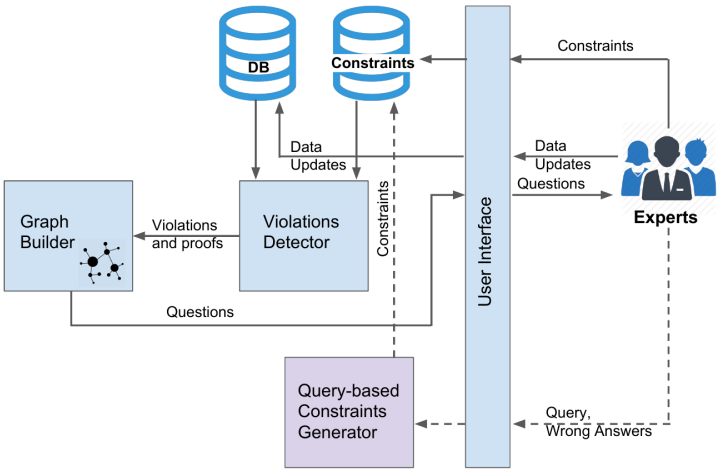


Figure 3.8: DANCE framework architecture presented in [4]

Detector component detects *Violations* of constraints and *Proofs* which are the suspicious tuples. Then, based on the *Violations* and *Proofs* found, the *Graph Builder* component builds a graph with suspicious and edge weights. After this stage, the *Graph Builder* runs the PageRank like algorithm to assign weights to the nodes (i.e., suspicious tuples) based on the edges. The user, an *Expert*, is then required to provide feedback by repairing the most weighted tuple. After each new *Data Update* (i.e., repair) provided by the *Expert*, the system updates and recomputes the node weights in order to get the next most weighted tuple to present to the user.

The alternative flow of this system (dashed lines in Figure 3.8) provide the support to clean the incorrect data identified by the user in a Query Oriented Cleaning scenario. For each tuple in the result of a given query posed to the database, that contains incorrect data, the *Query-based Constraints Generator* module converts it into a constraint that is added to the *Constraints* set. Then, the main flow of this system (continuous lines in Figure 3.8) is re-executed by computing the violations, generating the graph and asking the user for feedback on the most weighted tuple in order to resolve the violations to those constraints i.e., clean the incorrect data found by the user.

3.4.5 FALCON

FALCON [44] proposes a new data cleaning system based on SQL update statements (i.e., SQL update queries in the author's terminology) over a relational database table. The main challenge is to find the minimum set of SQL update statements that repair the largest number of data errors. For that purpose, first the user is requested to repair a cell of his/her choice (e.g, change value N.Y. in the Laboratory column of table to New York) and then FALCON tries to find the most general SQL update statement based on the user modification (e.g., UPDATE Tdrug SET Laboratory = "New York" WHERE Laboratory = "N.Y.")

In Figure 3.9, we present the FALCON workflow. First FALCON requests the user to provide a repair Δ over a cell in a data table T using the *Falcon Table UI* which presents a cell editable table to the user. Based on Δ, the FALCON *Rule Engine* component generates a set of Structured Query

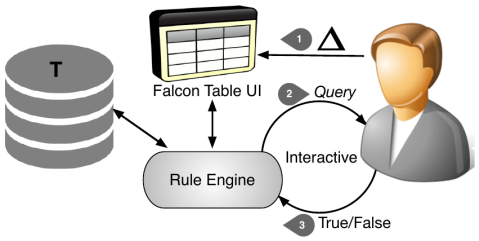


Figure 3.9: FALCON workflow presented in [44]

Language Update (SQLU) statements. It then asks the user to verify one of the statements Q taken from the SQLU set. If the user accepts the statement Q (i.e., declare the statement True or False), the *Rule Engine* applies it to the table T and continues asking the user to validate SQLU statements until there are no more generated statements to be verified or until the user has exceeded its budget for statement verification. When the *Rule Engine* stops asking to validate SQLU statements, then the process repeats by presenting to the user the *Falcon Table UI* containing the new instance of the data table T that resulted from applying the SQLU statements verified by the user. Then, if T still contains data quality problems, the user performs a new manual repair Δ over T.

In order to find SQLU statements, FALCON uses a lattice graph to search for SQLU statements that contain a repair Δ over T. The lattice graph is constructed based on the intuition that one SQLU statement Q is contained in another statement Q', i.e., Q' $(Q < Q')$, if the result of applying Q is a subset of applying Q' for any instance of the database table T. Having a lattice graph, the main problem for the authors was to decide which queries to present to the user and in which order. The authors proposed two algorithms for lattice search: One-Hop Search algorithm and Multi-Hop Search algorithm. One-Hop Search navigates from a node to its neighbors, searching can be Breadth First Search (BFS) or Depth First Search (DFS) based. The Multi-Hop Search algorithm is based on binary search and attribute correlations that prefer nodes that repair a large number of tuples.

3.4.6 ICARUS

ICARUS [90] is a framework that helps the user to infer new values for a database with missing values. This framework presents to the user a data matrix that may contain data from multiple relations so that she can fill empty cells with a value. After the user assigns a value, ICARUS generates all possible update rules that generalize on that value (similar to SQLU statements in FALCON) and then asks her to select the most general and valid rule from those. By presenting missing and complete data in the data matrix and by generating those rules selected by the user, ICARUS aims at minimizing the user effort while achieving high levels of data quality (that are not possible to be achieved using only automatic data cleaning approaches).

The ICARUS workflow is composed of two main components: (i) the *Generate Informative Subset* that generates subsets of data from a database in matrix format, and (ii) the *Generalize update to rules* that generates update rules based on a user inferred value in the data matrix. In Figure 3.10, the

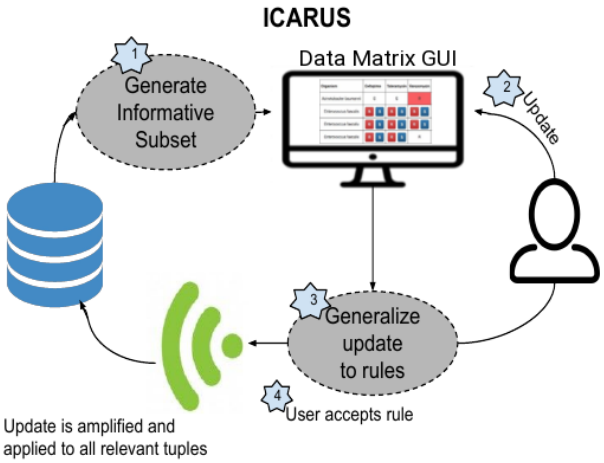
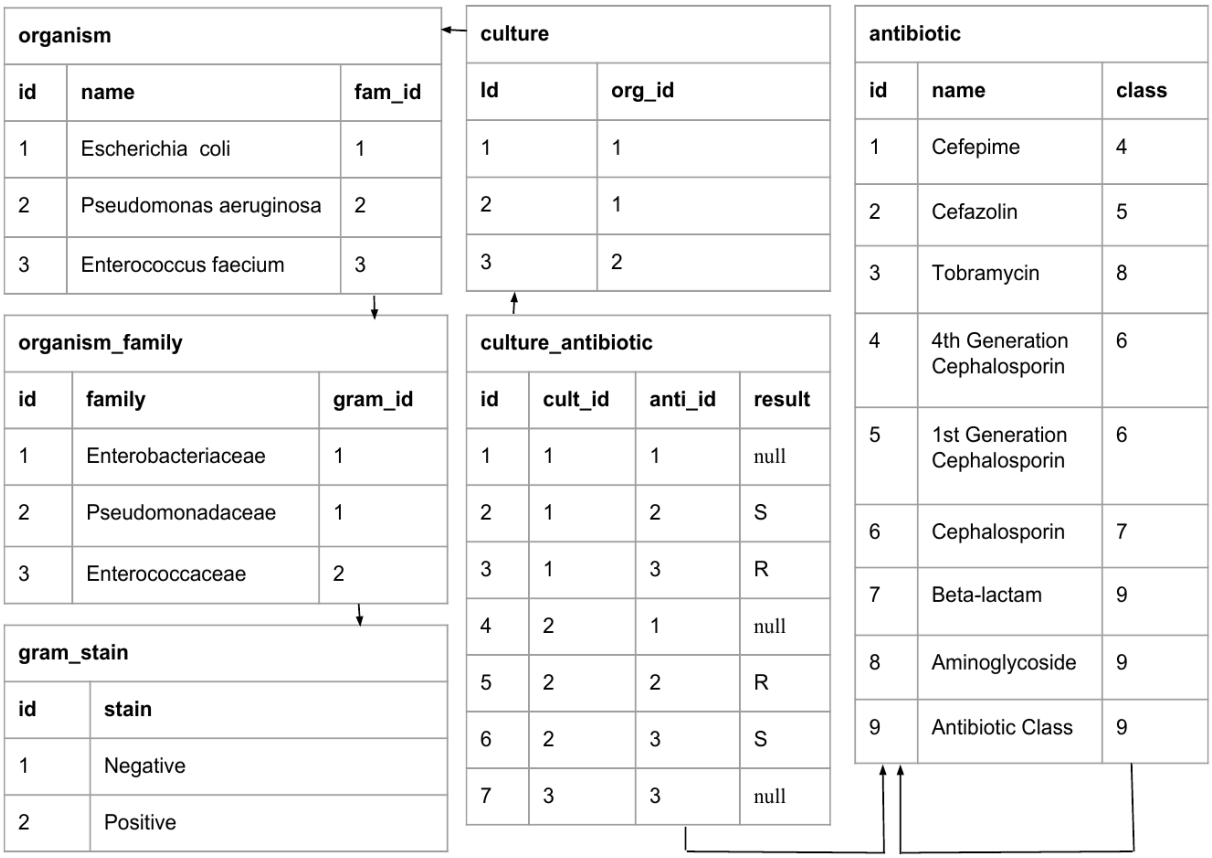


Figure 3.10: ICARUS workflow, modified from the presented in [90]

workflow is presented: for a database, the *Generate Informative Subset* generates a data matrix that is presented to the user in a Graphical User Interface (GUI). The user applies an *Update* to the data, which means she infers a value into an empty data matrix cell. Then, based on the *Update*, the *Generalize update to rules* generates update rules that are verified by the user. When the user selects an update rule, it is applied to the database.

The subsets of data from a database are generated in matrix format because it is more suitable for the user to inspect and complete data. The authors describe what is the best subset in terms of an optimization problem that minimizes the number of iterations i.e., the number of subsets that have to be generated for the user to infer missing values. This problem is a NP-complete problem called the *maximal weighted set cover* problem [38]. Due to this complexity, the authors use sampling-based techniques to generate the subsets. For that, they take into account the similarity between columns (user defined or using CORDS [47] that automatically finds correlations among columns) and the impact (i.e., number of values in the database affected by a user rule). The information entropy [62] for each row and column is also used to promote diversity (i.e., statistic independence) between rows and columns which is more important in later iterations where the user will create rules with more conditions (i.e., specialized rules). As an example consider Figure 3.11 presenting a microbiology culture database that contains: the *culture_antibiotic* table that reports if an organism growing in a culture is resistant (R) or sensitive (S) to particular antibiotics, the *culture* table that provides information of which organisms were in a particular culture, the tables *organism*, *organism_family* and *gram_stain* that describe the organisms in a hierarchy of organism name, family and stain, and table *antibiotic* that describes an antibiotic with name and class. In Figure 3.12, we present the data matrix returned by *Generate Informative Subset* for the microbiology culture database.

To generate update rules based on a user inferred value (*Update*), the authors implemented two types of rules that can generalize across multiple relations: the independent rules and the dependent rules. *Independent rules* are based on joins between relations. Intuitively, when we join the table where the value is inferred by the user (e.g., *culture_antibiotic* table in Figure 3.11) with other tables (e.g., *organism_family* table and *gram_strain* table in Figure 3.11) we can access higher levels in an existing



organism

id	name	fam_id
1	Escherichia coli	1
2	Pseudomonas aeruginosa	2
3	Enterococcus faecium	3

culture

Id	org_id
1	1
2	1
3	2

antibiotic

id	name	class
1	Cefepime	4
2	Cefazolin	5
3	Tobramycin	8
4	4th Generation Cephalosporin	6
5	1st Generation Cephalosporin	6
6	Cephalosporin	7
7	Beta-lactam	9
8	Aminoglycoside	9
9	Antibiotic Class	9

organism_family

id	family	gram_id
1	Enterobacteriaceae	1
2	Pseudomonadaceae	1
3	Enterococcaceae	2

culture_antibiotic

id	cult_id	anti_id	result
1	1	1	null
2	1	2	S
3	1	3	R
4	2	1	null
5	2	2	R
6	2	3	S
7	3	3	null

gram_stain

id	stain
1	Negative
2	Positive

Figure 3.11: Example of a microbiology culture database presented in [90]

culture_id	organism	Cefepime	Cefazolin	Tobramycin
1	Escherichia coli	null	S	R
2	Escherichia coli	null	R	S
3	Pseudomonas aeruginosa	R	null	null

Figure 3.12: Example of a data matrix for the microbiology culture database presented in [90]

data hierarchy (e.g., culture organism name $>$ organism family $>$ gram stain from the microbiology culture database) which can be used to produce even more general update rules based on conditions that include higher hierarchy levels. An independent rule example for the microbiology culture database would be: *all organisms belonging to family EnteroBacteriaceae are sensitive to Cefeprime*. *Dependent rules* generalize the inferred value to other tuple values in the matrix by considering those values that belong to similar columns (similarity used in the generation of subsets). A generated dependent rule example based on column similarity for the microbiology culture database would be: *organisms sensitive to Cefazolin are also sensitive to Cefeprime*.

3.4.7 Discussion

In LLUNATIC, the user aims at resolving conflicts left by automatic generated repairs for violations of data quality constraints. The user provides a value for cells marked with lluns.

In another line of work, Guided Data Repairs take advantage of the user to replace heuristics for selecting data repairs to solve CFDs violations while in Continuous Data Cleaning the user has to train a model that to solve a given violation decides whether to repair the data or to specialize the violated Functional Dependency (FD). DANCE, FALCON, and ICARUS aim at achieving high data quality by refusing the automatic selection of data repairs. Thus, they delegate data repairing exclusively to the user. DANCE tries to clean data by relying only on the user to produce data repairs. It minimizes the user effort by guiding her through the tuples that are the most likely to need data repairs. FALCON and ICARUS allow the user to first edit a cell and then generate general update statements based on the user modification. FALCON supports the edition of existing missing or incorrect values and focus on minimizing the user effort in generating updated statements, whereas ICARUS supports only the edition of missing values but obtains better performance and also focus on selecting the initial data to present to the user.

Related Work	**User Involvement**	**Automatic selection of data repairs**	**Data Quality Rules**	**Datasets**
LLUNATIC	Resolve conflicting data repairs	Yes	EGDs	Hospital and Customers
Guided Data Repairs	Select data repairs	Yes	CFDs	Hospital and UCI adult
Continuous Data Cleaning	Select repair types	Yes	FDs	Finance, Vehicles, Veterans and Linked Clinical Trials
DANCE	Provide data repairs	No	Integrity Constraints	World Cup and Flights
FALCON	Provide data repairs and select update statements	No	No	Soccer, Hospital, BUS, and DBLP
ICARUS	Provide data repairs and select update statements	No	No	Microbiology, IMDB, and Hospital

Table 3.3: Summary of related work in user involvement in data cleaning.

In Table 3.3, we present the summary of related work that explores the user involvement in data

cleaning and the datasets used in each work. Note, that even if some datasets are common, their errors are not since most of these works address different data quality problems.

So far, in the context of the user involvement in a data cleaning process, the user has been proposed to help resolving data quality constraint violations or to generate update statements (which can also be constraints) based on edited cells. The importance of user feedback to achieve high levels of quality is shared across all these works for specific tasks, e.g., repairing missing values. Nevertheless, the user has not been involved in a data cleaning process where data cleaning programs are based on data transformations, which is the most common approach in companies but not in research. Moreover, user involvement was not explored when continuous program refinements are applied through an iterative process. Note that Continuous Data Cleaning addresses only stream data by proposing to specialize FDs.

Chapter 4

Acronym Expansion

In this chapter, we report the work concluded in the context of this thesis for acronym expansion. The work related to acronym expansion was in collaboration with João Casanova, that was a MsC student at Instituto Superior Técnico [1].

This chapter describes the following contributions:

- The **end-to-end Acronym eXpander (AcX) system** accepts a text document as input and outputs a list of acronym-expansion pairs for the acronyms found in the document, whether or not the expansions are in the document. As far as we know, AcX is the first open source and extensible system for acronym expansion that allows both the mixing and the combination of different inference modules.

  AcX is easily extendible to other languages. We already have versions in English, French and Portuguese. Students with no previous background in natural language processing or machine learning have already done so for French and Portuguese.

- A **benchmark of in-expansion techniques (in-expansion benchmark)**. We make use of four biomedical datasets previously proposed in the literature (i.e., Medstract [48], Schwartz and Hearst [48], BIOADI [48], and Ab3p [48]) and one of the biggest sentence based datasets from the scientific domain (i.e., SciAI [84]). Additionally, we have created a new dataset composed of Wikipedia documents from the *Computing* category. We calculate the precision, recall, and F1-measure for the extraction of both acronyms and acronym-expansion pairs. In addition, we measure training and execution times. Additionally, we have implemented, integrated and evaluated two rule-based (Schwartz and Hearst [97], MadDog [110]) and two machine-learning (based on SciBERT [9] used in [99], and SciDr [99]) acronym-expansion extraction techniques. Our experimental results show that there is not a single best technique for every dataset. Moreover, rule-based techniques overall achieve better precision while machine-learning techniques achieve better recall.

[1] João Casanova successfully finished his MsC thesis in acronym and in-expansion extraction co-supervised by the candidate. Specifically, the candidate did the first version of the in-expansion benchmark and of the user generated dataset described in the following sections. João Casanova extended the mentioned benchmark with new metrics (Cohen Kappa, and glossary level precision, recall and F1-measure), implementations of SciDr and MadDog in-expanders and Schwartz and Hearst implementation in python (before we had the original one in Java). He also extended the user generated dataset with additional articles (around 100), and parsed a recent dataset, SciAI. He also ran the experiments for the in-expansion benchmark.

- A **benchmark of out-expansion techniques (out-expansion benchmark)**. We evaluate out-expansion techniques on three datasets from different domains previously used in related work that contain documents (i.e., MSH [85], SciWISE [85], and CS Wiki [107]) and one that is constructed from independent sentences from the scientific domain (i.e., SciAD [111] revised by Egan and Bohannon [30]).

  Some of the out-expansion techniques we consider have already been proposed in related work: Classic Context Vector [2, 61, 85], Surrounding Based Embedding [61], Thakker et al. [107], and Unsupervised Abbreviation Disambiguation [21]. Most recently, competitors of the SDU@AAAI competition [111] mainly used pre-trained language models based on Transformer neural networks like BERT [28] and SciBERT [9]. For purposes of out-expansion, we have adapted and evaluated: (i) the out-expander of the MadDog system [110]; (ii) SciDr [99] (which was originally hard coded to apply to the sentences of the SDU@AAAI competition, but we have extended it to multiple datasets of documents); (iii) LUKE (Language Understanding with Knowledge-based Embeddings) [115] originally for Entity Disambiguation, but we extended it to use acronyms and expansions; and (iv) techniques from Natural Language Processing (i.e., Term Frequency–Inverse Document Frequency [102], Latent Dirichlet Allocation [12], Doc2Vec [58], and Sentence Bidirectional Encoder Representations from Transformers (SBERT) [91]).

  We have also embedded the outputs of a variety of out-expansion techniques as features for machine learning classifiers. The net result is that AcX is an extensible system able to create a new set of out-expansion pipelines out of combinations of user-chosen techniques.

  We evaluate the techniques using out-expansion accuracy (most common for these kind of disambiguation problems) and the macro F1-measure (used in the SDU@AAAI competition). We also measure execution times for both technique training and document processing. Our results show that ensembler techniques give the best accuracies and macro F1-measure. Cossim with SBERT and SciDr out-expander are the indivudal techniques that score best. At a slight loss in accuracy, Classic Context Vectors, or even Cossim or SVMs with Doc2Vec are almost 10 times fast on average.

- A **benchmark of end-to-end acronym expander systems (end-to-end benchmark)**. We create the first end-to-end dataset of human-annotated documents that includes both in- and out-expansions. We have built a human-generated end-to-end benchmark because previous annotated datasets used automatic mechanisms to identify acronyms and those automatic techniques are neither accurate nor complete. Thus, human annotation offers a kind of gold standard. We use this dataset to test AcX as well as the state-of-the-art acronym expansion systems. We use all English Wikipedia documents to train it. We compare the MadDog system [110] against different pipelines of AcX. We also compare AcX's performance to that of human annotators. Those systems are evaluated using the precision, recall, and F1-measure for the acronym-expansion pairs returned by a system for each document. We also measure execution times. Our best system pipeline correctly expands most acronyms (54.97% of F1-Measure, only about 27% worse than

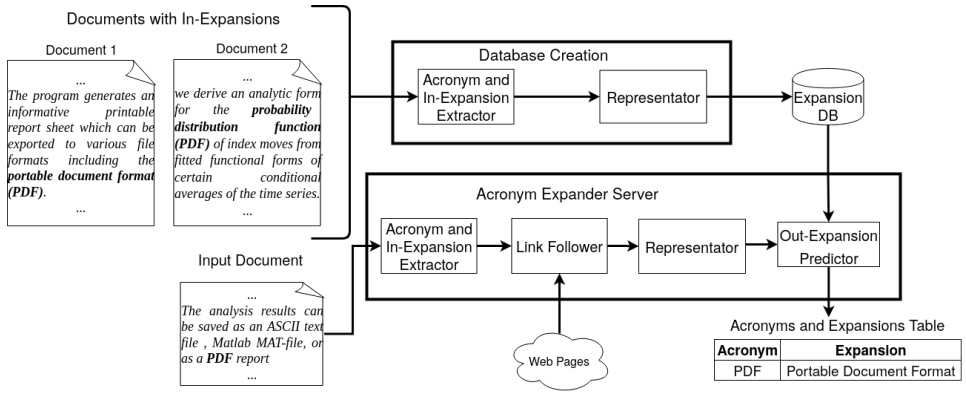


Figure 4.1: Acronym eXpander (AcX) system. The top stream denotes the creation of the Expansion Database that associates each <acronym, in-expansion> pair with some representation of the document(s) where that pair was found. The bottom stream shows the processing of an input document d by combining acronym in-expansion when possible and a representation of d. For an acronym A with no expansion in d, the representation of d is compared with the representations in the Expansion Database of documents containing A to find the context-appropriate expansion.

humans) taking on average less than 2s per document.

- **A link follower component** within AcX system that infers expansions based on the links of unexpanded acronyms in input documents. Link following improves the F1-measure of the best AcX pipeline identified by our end-to-end benchmark. Our results show that our best system pipeline slight improves, +0.37 of F1-measure, when this component is used.

The work concluded in this thesis for acronym expansion led to: (i) a full paper published to the proceedings of the Very Large Data Bases (VLDB) Endowment about our end-to-end system [81], Acronym eXpander (AcX), and the three benchmarks; (ii) a full paper presented in the AAAI-21 Workshop on Scientific Document Understanding (SDU@AAAI) reporting our participation in the competition for out-expansion [80]; (iii) one poster presented in three events: the IST PhD Open Days 2019[2], in the IST Taguspark Thesis Showcase – Spring Edition 2019[3], and in the 9th Lisbon Machine Learning School[4] and (iv) two demonstrations: one at IST Taguspark Thesis Showcase – Spring Edition 2019, and another at the 9th Lisbon Machine Learning School.

This chapter is organized as follows: Section 4.1 describes our proposed Acronym eXpander (AcX) system and its components. The next three sections (Sections 4.2, 4.3 and 4.4) describe the proposed benchmarks and the corresponding analyses of the results obtained from the experiments performed in the context of each benchmark.

4.1 AcX: an End-to-end Acronym eXpander System

The AcX system (see Figure 4.1) consists of:

[2] http://phdopendays.tecnico.ulisboa.pt/
[3] https://tecnico.ulisboa.pt/pt/eventos/mostra-de-teses-2019-edicao-primavera/
[4] http://lxmls.it.pt/2019/

(i) A *Database Creation* process which generates an *Expansion Database*[5] that contains documents, acronyms and their corresponding in-expansions. The Expansion Database also associates each <acronym, in-expansion> pair with a *representation* of the document where that acronym and in-expansion were found. The representation characterizes the content of the document in some summary form. To support other domains and languages, we pass documents in the desired domains/languages to the Database Creation process. In Section A.2 of Annex A, we present additional detail about data structures used in AcX database.
(ii) The *Acronym Expander Server* that accepts one document at a time from a user and outputs a list of acronyms found in the input document and the corresponding expansions found by the system (whether as in-expansions or as out-expansions).

For each document with in-expansions, the *Database Creation* process runs the following pipeline:

1. an *Acronym and In-Expansion Extractor* obtains the <acronym, expansion> pairs from the document using only within-document evidence.

2. a *Representator* (there are many possible representators e.g., Latent Dirichlet Allocation that output topics) maps the document to a document representation that holds document contextual information.

3. the *Expansion Database* stores the in-expansions, acronyms, and document representations on disk, currently SQLite [45].

Given a new input document d supplied by a user, the *Acronym Expander Server* executes the following pipeline:

1. applies the *Acronym and In-Expansion Extractor* used to build the Expansion Database to extract all the acronyms having expansions in the input document d.

2. when d contains links to web pages then those page texts are extracted and inspected to find the expansion for acronyms whose expansions are not found in d (*Link Follower*).

3. utilizes the same *Representator* (say, topics from Latent Dirichlet Allocation) used to characterize each document in the Expansion Database to map d to a document representation.

4. for each acronym A having no in-expansion in d, the server runs the *Out-Expansion Predictor* to choose a context-appropriate out-expansion. Formally, an expansion E is selected for an acronym A in d if the representations of the documents $doc(A, E)$ with expansion E share more characteristics with the representation of d by some criteria (e.g., closest cosine similarities or labeled by some machine learning classifier for A) than the documents in $doc(A, E')$ for every alternative expansion E'. Thus, for example, if the context of d is publishing, then "PDF" should likely expand to "Portable Document Format" but if the context of d is probability or statistics, then "PDF" should expand to "probability distribution function".

[5] When benchmarking, the expansion database will provide us with both a training set and a test set.

For a language other than English, the in- and out-expansion techniques should be tuned to the new language. They may benefit from changing preprocessing steps such as tokenization for the new language or from adopting a language model trained on the new language or even adopting a multilanguage model.

4.1.1 Acronym and In-Expansion Extraction

Acronym and in-expansion extraction can use rule-based or machine learning technique. Currently, we have integrated the in-expander implementations of Schwartz and Hearst [97], MadDog in-expander [110], SciBERT [99] and SciDr in-expander [99]. In our rule-based implementations (i.e. Schwartz and Hearst [97] and MadDog [110]), we used roughly the following three-step process as described in [75]:

1. *Acronym extraction:* identifies acronyms in a document, e.g., PDF in Figure 4.1. We modified Schwartz and Hearst [97] to find candidate acronyms even when there is no expansion found in a given document. The technique excludes tokens in which all alphabetic characters except the first character are lower case. We also reject acronyms of two characters where the first is a letter and the second is a dot "." to avoid person names.

2. *Candidate expansion extraction:* builds candidate pairs of acronyms and possible in-expansions <acronym, expansion> from information in the document, e.g., <PDF, formats including the portable document format> from Document 1 in Figure 4.1.

3. *Candidate refinement:* evaluates each candidate pair using a variety of heuristics (e.g., find the shortest expansion that matches the acronym) to obtain a final in-expansion for each acronym that has at least one candidate in-expansion within the document, e.g., portable document format from <PDF, formats including the portable document format>.

For the in-expanders of SciBERT and SciDr, the extraction of acronyms and expansions is formalized as a sequence tagging problem where each token can have one of three tags: (i) a token in an acronym (e.g., CD in CD-ROM), (ii) a tokenword in an expansion, or (iii) other token. For example, from Document 1 in Figure 4.1, PDF would be tagged as an acronym token, each token portable, document, and format would be tagged as a token in an expansion. The remaining tokens in Document 1 would have the "other token" tag. AcX appropriately tags the acronym-expansion pairs and other tokens in the training data, then builds a machine learning model on the tagged data. The output of such machine learning models is then converted to acronym-expansion pairs by matching the acronym characters against expansions.

Our system supports ensemble in-expansion through SciDr. That ensemble technique can be easily extended to include additional in-expansion techniques.

4.1.1.1 Link Follower

For an input document d containing hyperlinks, the *Link Follower* component follows those links to try to find the expansions from documents pointed to by d. This module is used after executing the acronym and in-expansion extraction algorithm for the acronyms with no expansion found in text. For

each A acronym having no in-expansion, the Link Follower is constituted by the following steps, until an expansion is found:

1. searches in the *title* attribute of the hyperlink a HyperText Markup Language (HTML) tag for the expansion. For example, given the following tag *LEED* the expansion "Leadership in Energy and Environmental Design" would be extracted for acronym LEED.
2. resolves relative HTML Uniform Resource Locator (URL) links to global ones. For example, if no expansion were found in the example above with a local URL, the resolved URL would be `https://en.wikipedia.org/wiki/Leadership_in_Energy_and_Environmental_Design`.
3. executes the same in-expander technique used by the acronym and in-expansion extractor.

Given a title, the acronym is placed inside parenthesis and appended to the end. Then, the in-expander technique is executed on the title in order to extract a possible expansion.

4.1.2 Representator

Representors in the AcX system summarize documents in order to capture knowledge about their semantics. Although AcX supports sentence-level out-expansion techniques, using the whole document is more effective than using just parts of the text because the whole document captures the overall context better.

Some representators assign a set of topic termsdescription to a document. If two documents have many topicrepresentative terms in common, then they are considered to be semantically related.

Other representators use embeddings [58] to characterize a document. An *embedding* is a vector of real numbers in a high dimensional space. Embedding techniques map an object encoded in a one-hot representation, a very sparse and high dimensional vector of binary values, into a very dense and lower dimensional vector of real values (i.e., embedding). A small distance between embedding vectors suggests document similarity.

AcX encloses several techniques that can semantically represent an entire set of documents that contain the same expansion for a given acronym. Specifically, let $docs(A, E)$ denote the set of full document texts in which a given acronym A is defined by a single expansion E (e.g., all documents in which acronym PDF is explicitly expanded as portable document format):

Here are some representations of such a collection of documents:

- *Classic Context Vector (CCV)* [2], represents an expansion E by the set of words in $docs(A, E)$ along with their counts.
- *Document Context Vector (DCV)* (our variation of context vector), builds on context vector, however it represents each document $d \in docs(A, E)$ individually by the set of word occurrences in d. For example, the word occurrences corresponding to Document 2 in Figure 4.1 would contain among others the values $\{$of: 3$\}$, $\{$the: 2$\}$, $\{$derive: 1$\}$, $\{$analytic: 1$\}$, $\{$form: 1$\}$.

- *Term Frequency–Inverse Document Frequency (TF-IDF)* [102], weights each term t in each document $d \in docs(A, E)$ highly if it is found frequently in d and infrequently in the entire document corpus, thus permitting the characterization of each document by its highly weighted terms. For example, the TF-IDF score for the word the in Document 2 in Figure 4.1 is $\frac{2}{27} \cdot log(\frac{2}{2}) = 0$ because this word appears in both documents.

- *Latent Dirichlet Allocation (LDA)* [12] assigns topics to documents using a Dirichlet probabilistic model. For example, Document 2 in Figure 4.1 could be represented by the following topics: topic1={{analytics: 0.7}, {series: 0.3}} and topic2={{functional: 0.8}, {form: 0.2}}.

- *Doc2Vec* [58] is a document embedding technique based on Word2Vec [68] which assigns vectors to words in such a way that words that appear in the same context have a high cosine similarity. For example, the words functional and conditional would be assigned similar vectors. Thus, using the principles of Word2Vec, Doc2Vec assigns vectors to entire documents. For example, documents 1 and 2 in Figure 4.1 would be assigned mutually distant vectors.

- *Sentence Bidirectional Encoder Representations from Transformers (SBERT) [91]* constructs sentence embeddings that can be compared to determine sentence similarity. AcX splits the input document text to fit into the SBERT input limit (e.g., 384 tokens), and then we average the resulting embedding vectors to get a document representation.

4.1.3 Out-Expansion Predictor

To choose an out-expansion for an acronym A in an input document d having no expansion for A, the Out-Expansion Predictor component considers each candidate out-expansion E for A and compares d to some representation of $d' \in docs(A, E)$.

In the case of Classic Context Vector (CCV), we compare d with the vector representation of $docs(A, E)$. For the remaining techniques, we compare d with each document representation of $d \in docs(A, E)$.

Using cosine similarity, the Out-Expansion Predictor will choose an out-expansion E over a different expansion E' if any document $d \in docs(A, E)$ is more similar to d than all $d''' \in docs(A, E')$.

A classical similarity technique is cosine similarity, but the AcX system also supports classification-based approaches that work as follows. Consider all the documents, denoted $alldocs(A)$ containing in-expansions of acronym A. Some documents in $alldocs\ (A)$ have an in-expansion of $E1$ for A, some have $E2$ for A and so on. Given the representations of documents in $alldocs(A)$ as features and the expansions ($E1$, $E2$, etc) as labels, the out-expansion problem becomes a machine learning classification problem. When a new document d is given to AcX, the representation of d is passed as input (i.e., features) to the classifier which labels d with an expansion.

The classifiers we support so far are:

- *Support Vector Machines (SVMs)* [24] fit a hyper-plane that optimally separates binary labeled data in the feature space. Non-binary classification is performed by a "one-vs-all" technique where

a binary SVM classifier predicts with a certain probability if an input document belongs to a particular class (where each class corresponds to a particular expansion). The class (and therefore expansion) with the highest probability is selected. We used the LibLinear [33] implementation included in sckit-learn toolkit [77].

- *Logistic Regression (LR)* [51] fits a logistic function to classify binary classes (again a class corresponds to an expansion). Non binary classification is again performed by a "one-vs-all" technique. We used the LibLinear [33] implementation included in scikit-learn toolkit [77].

- *Random Forests (RF)* [15] fit a particular number of decision trees (default 100) trained on randomly selected samples. There will be one random forest per acronym A. The representation of a document having no in-expansion for A will be input to the random forest. Each tree will predict one expansion with some probability. The random forest selects the class whose average probability is the highest. We used the scikit-learn [77] implementation.

In addition to these classifiers, for evaluation purposes or for anyone who wants to try other techniques, AcX supports the following additional techniques from related work: Surrounding Based Embedding (**SBE**) [61], Thakker et al. [107], Unsupervised Abbreviation Disambiguation (**UAD**) [21], the SciDr out-expander (**SciDr-out**) [99], the MadDog out-expander (**MadDog-out**) [110], and **LUKE** [115], a state-of-the-art technique for Entity Disambiguation. For UAD, SciDr-out and MadDog-out, AcX performs sentence segmentation and, given the results from each sentence, decides which expansion to assign to the text. For UAD, we select the most frequent predicted expansion among the sentences in the document.

We have extended SciDr-out to consider all the sentences containing the acronym A instead of just one sentence as in SciDr-out's original implementation. SciDr-out associates an acronym with its possible expansions concatenated together. The system then finds the substring of that concatenated string with the highest probability and outputs that as the expansion. For example, the concatenated expansion of "PDF" might be "probability density function portable document format". Depending on the contents of some input document d containing "PDF", SciDr-out will choose some substring of that concatenated expansion.

We have extended MadDog-out to enable it to train in new documents, instead of using only their original machine learning models. MadDog-out processes the last sentence of any document containing acronym A to determine the most likely expansion.

For LUKE, we had to modify the internals to work with acronyms and expansions. We use their pretrained model and perform fine-tuning in our training data using the procedure described by the authors in [115], except that we allow the entity embeddings (now expansion embeddings) to be updated during training. This modification allows the generation of embeddings for expansions out of the original model vocabulary.

4.2 In-expansion Benchmark, Evaluation and Results

We describe our benchmark of in-expansion techniques in Section 4.2.1 and evaluate state-of-the-art techniques on this benchmark in Section 4.2.2.

4.2.1 A Benchmark of In-expansion Techniques

This section describes the benchmark we developed to evaluate in-expansion techniques. Section 4.2.1.1 details the datasets used in this benchmark. Section 4.2.1.2 lists the in-expansion techniques that we implemented for this benchmark. Section 4.2.1.3 defines the metrics that we used to evaluate the in-expansion extraction techniques.

4.2.1.1 Datasets

The datasets included in this in-expansion benchmark are:

Medstract: This dataset is composed of 199 randomly selected MEDLINE[6] abstracts from the results of a query on the term ”gene”. The abstracts were manually annotated and then the annotations were corrected and improved by Schwartz and Hearst [97], Ao and Takagi [3], Pustejovsky et al. [86], Yarygina and Vassilieva [117] and Doğan et al. [48]. We use the last revised version of Doğan et al. [48] that contains 159 acronym-expansion pairs.

Schwartz and Hearst: This dataset consists of 1,000 randomly selected MEDLINE abstracts from the results of a query on the term ”yeast”. The abstracts were manually annotated by Schwartz and Hearst [97] and revised by Doğan et al. [48]. The revised version that we use contains 979 acronym-expansion pairs.

BIOADI: This dataset contains 1,201 abstracts from the BioCreative II gene normalization dataset. The dataset was original annotated by Kuo et al. [56] and revised by Doğan et al. [48]. It contains 1,720 acronym-expansion pairs.

Ab3P: This dataset results from the random selection of MEDLINE 1,250 abstracts. The dataset was manually annotated by Sohn et al. [100]. We use the revised version of Doğan et al. [48] that contains 1 223 acronym-expansion pairs.

SciAI: This dataset results from processing 6,786 English arXiv[7] papers. Those papers were split into sentences and sent to Amazon Mechanical Turk (MTurk) to be annotated by humans, resulting in 9,775 acronym-expansion pairs. This dataset was annotated for both acronyms and acronym-expansion pairs. The final dataset has 17,506 sentences, where 1% do not contain acronyms and 24% do not contain expansions. We use the SDU@AAAI competition [111] version[8] that was initially proposed by Veyseh et al. [84].

[6] https://www.nlm.nih.gov/bsd/medline.html
[7] https://arxiv.org/
[8] https://github.com/amirveyseh/AAAI-21-SDU-shared-task-1-AI

End-to-end: We developed a dataset that consists of 163 English Wikipedia documents randomly selected from the *Computing* category[9] in Wikipedia. It contains 1,139 acronym-expansion pairs. Although intended to evaluate an end-to-end acronym-expander system, for this in-expansion benchmark in particular, we consider only the acronym-expansion pairs with expansion in text. (Later, in Section 4.4.1, we use the whole set of acronym-expansions pairs to evaluate end-to-end systems.) Each document was annotated by two computer science students who volunteered for the task. Each student annotated at least two documents. During the annotation process, each student identified each acronym in the document and mapped it to an expansion. Each acronym-expansion pair was labeled by the annotators, indicating whether the expansion was present in text. Any conflict between annotators was manually resolved by the authors. The Inter-Annotator Agreement (IAA) among each annotators (excluding the third annotator, the reviewer) using Krippendorff's alpha [54] with the Measuring agreement on set-valued items (MASI) distance metric [76] is 0.68 for in-expansion pairs and 0.33 for out-expansion pairs. In a hypothetical scenario, if both annotators had given the same acronym-expansions, then the score would be 1. In this case, the human annotators disagree on out-expansions more often than on in-expansions. This is unsurprising because out-expansion requires consulting additional text sources other than the document at hand, while for in-expansion the text provided is enough. In Section A.1 of Annex A, we present additional detail about the process used to create this dataset.

4.2.1.2 In-expansion techniques

This benchmark includes the following in-expansion techniques (that are supported by our AcX system described in Section 4.1):

Rule-based: Schwartz and Hearst (**SH**) [97] technique and the MadDog [110] in-expansion (**MadDog-in**) technique which builds on the Schwartz and Hearst algorithm.

Machine Learning: SciBERT based technique used in [99] and the SciDr [99] in-expansion (**SciDr-in**) technique which ensembles SciBERT models and a rule-based technique based on SH with Conditional Random Fields. Moreover, we consider models used by these machine learning techniques that are trained *with external data* besides the individual training sets of each dataset. The external data is composed of Medstract, Schwartz and Hearst, BIOADI, and Ab3P train sets if the test set is biomedical. For SciAI and End-to-end test sets, the external data consists of all train sets (i.e., biomedical datasets, SciAI, and End-to-end).

4.2.1.3 Performance metrics

Our benchmark uses the following metrics. The metrics apply to acronyms alone as well as to acronym-expansion pairs. The acronyms can be either in singular or plural form to be considered equal, and the expansions are equal if their lower case versions without dashes have an edit distance

[9] https://en.wikipedia.org/wiki/Category:Computing

less than 3 or if the first 4 characters of each word are equal. If the same acronym or pair appears several times in the same document, it is counted only once:

Acronym Pair Precision: the number of correctly extracted acronym pairs divided by the number of acronym pairs extracted by that technique over all documents. It will be calculated as:

$\frac{\#\ of\ correctly\ extracted\ acronyms\ pairs}{\#\ of\ extracted\ acronyms\ pairs}$.

Acronym Pair Recall: the number of correctly extracted acronym pairs divided by the number of distinct acronym pairs present over all documents. It will be calculated as:

$\frac{\#\ of\ correctly\ extracted\ acronym\ pairs}{\#\ acronym\ pairs\ in\ text}$.

Acronym Pair F1-measure: the harmonic mean of the *precision* and *recall* of the system. It will be calculated as:

$2 * \frac{Precision * Recall}{Precision + Recall}$.

Acronym Pair Cohen's Kappa: measures the agreement between the acronym pairs extracted by a technique and the actual acronym pairs present in text for a given number of documents in a dataset. The value of *kappa* [22] is calculated as $\frac{p_o - p_e}{1 - p_e}$, where p_o is the probability of agreement between an acronym-expansion extraction technique and the actual dataset and p_e is the probability of random agreement. This measure is particularly useful for unbalanced test sets, because it gives greater weight to infrequent acronym-expansions (for which the denominator will be smaller). To avoid treating acronyms and expansions with small differences as entirely different labels (i.e, full disagreement), we calculate kappa with respect to the *Jaccard* [49] distance.

Training time: Central Processing Unit (CPU) or Graphics Processing Unit (GPU) time in seconds to train the machine-learning models that are used by the in-expansion technique.

Execution time: CPU or GPU time in seconds that the in-expansion technique takes to extract acronym-expansion pairs from a document in the dataset.

By default (i.e., when not mentioned otherwise), acronym pair *Precision*, *Recall*, and *F1-Measure* are calculate on the *Document-Level* where every unique acronym-expansion pair (or acronym) extracted from each document is counted. Thus, an acronym-expansion pair that appears many times across documents in the corpus will be weighted more than one that appears less often. This is in line with the most recent efforts in in-expansion which aim to efficiently extract acronym-expansions from each individual document independently.

Additionally, we calculate the *Precision*, *Recall*, and *F1-Measure* alternative level, glossary. In the *Glossary-level*, an acronym-expansion pair (or just acronym) that appears in the text or is expanded multiple times in different documents in the corpus is counted only once. This follows the first measurements performed in-expansion techniques which aimed to create only a global glossary (dictionary) based on a set of documents.

To better explain the *Document* and *Glossary* levels, let us consider a dataset D that is composed of documents A and B. In document A there are five instances of the acronym-expansion pair x and

Acronym and In-expansion Technique	Acronym																			
	Medstract				SH				BIOADI				Ab3P				Averages			
	P	R	F1	K	P	R	F1	K	P	R	F1	K	P	R	F1	K	P	R	F1	K
SH	**100.00%**	**89.13%**	**94.25%**	**0.97**	**99.56%**	**81.13%**	**89.50%**	**0.91**	**99.06%**	**79.58%**	**88.25%**	**0.88**	98.62%	**77.66%**	**86.89%**	**0.89**	**99.31%**	**81.88%**	**89.72%**	**0.91**
MadDog	**100.00%**	63.30%	77.33%	0.77	96.64%	51.80%	67.45%	0.77	97.99%	55.19%	70.61%	0.74	**99.16%**	64.31%	78.02%	0.85	98.45%	58.65%	73.35%	0.78
SciBERT	81.25%	56.52%	66.67%	0.68	85.56%	70.50%	77.47%	0.77	88.06%	73.91%	80.37%	0.76	85.99%	71.93%	78.34%	0.78	85.22%	68.22%	75.71%	0.75
SciBERT with External Data	86.84%	71.74%	78.57%	0.76	88.16%	77.70%	82.60%	0.82	90.61%	78.45%	84.09%	0.82	87.46%	76.02%	81.34%	0.81	88.27%	75.98%	81.65%	0.80
SciDr	93.33%	60.87%	73.68%	0.73	87.56%	60.79%	71.76%	0.74	92.11%	64.08%	75.58%	0.75	89.25%	67.85%	77.09%	0.79	90.56%	63.40%	74.53%	0.75
SciDr-in with External Data	92.11%	76.09%	83.33%	0.83	93.13%	78.05%	84.93%	0.86	93.31%	79.21%	85.69%	0.84	89.08%	71.11%	79.09%	0.81	91.91%	76.12%	83.26%	0.84

Table 4.1: In-expansion techniques Precision, Recall, F1-measures and Cohen's kappas (K) for acronym for each biomedical dataset (Medstract, SH, BIOADI, and Ab3p) and the averages.

Acronym and In-expansion Technique	Pair																			
	Medstract				SH				BIOADI				Ab3P				Average			
	P	R	F1	K	P	R	F1	K	P	R	F1	K	P	R	F1	K	P	R	F1	K
SH	**100.00%**	**89.13%**	**94.25%**	**0.97**	**96.03%**	**78.42%**	**86.34%**	**0.89**	**94.11%**	**75.61%**	**83.86%**	**0.83**	95.16%	**74.93%**	**83.84%**	**0.86**	**96.33%**	**79.52%**	**87.07%**	**0.89**
MadDog	93.10%	58.69%	72.00%	0.35	93.29%	50.00%	65.11%	0.47	87.58%	49.33%	63.12%	0.48	**95.37%**	61.85%	75.04%	0.64	92.34%	54.97%	68.82%	0.49
SciBERT	65.62%	45.65%	53.84%	0.27	72.37%	59.35%	65.21%	0.41	67.79%	56.90%	61.87%	0.36	74.27%	62.13%	67.66%	0.53	70.01%	56.01%	62.15%	0.39
SciBERT with External Data	76.32%	63.04%	69.05%	0.33	76.32%	67.26%	71.51%	0.42	74.45%	64.46%	69.10%	0.44	76.80%	66.76%	71.43%	0.54	75.97%	65.38%	70.27%	0.43
SciDr	80.00%	52.17%	63.16%	0.38	74.61%	51.79%	61.14%	0.4	76.90%	53.50%	63.10%	0.41	81.00%	61.58%	69.97%	0.54	78.13%	54.76%	64.34%	0.43
SciDr-in with External Data	92.11%	76.09%	83.33%	0.39	83.69%	70.14%	76.32%	0.49	86.19%	73.16%	79.14%	0.5	80.20%	64.03%	71.21%	0.52	85.55%	70.86%	77.50%	0.48

Table 4.2: In-expansion techniques Precision, Recall, F1-measures and Cohen's kappas (K) for pair for each biomedical dataset (Medstract, SH, BIOADI, and Ab3p) and the averages.

in document B two, with D having seven instances of x. Furthermore, a system for acronym-expansion extraction is run on dataset D and only extracts the five instances of x in document A. On the Glossary level, the number of correctly extracted pairs and the total number of extracted pairs are equal to one. Furthermore, the number of total acronym-expansion pairs present in D is only counted as one, because only unique pairs are considered. This results in a *pair precision* of $\frac{1}{1} = 1$ and a *pair recall* of $\frac{1}{1} = 1$. On the document level, the number of correctly extracted pairs and total number of extracted pairs is five. However, the number of total acronym-expansion pairs present in D is counted as seven, due to the five instances of x in document A and two instances in document B. This results in a *pair precision* of $\frac{5}{5} = 1$ and a *pair recall* of $\frac{5}{7} = 0.7$.

4.2.2 In-expansion Experimental Evaluation

In this section, we evaluate the in-expansion techniques using the benchmark presented in Section 4.2.1.

Setup. The in-expansion experiments were performed on a machine with an Intel® Core™ i5-4690K CPU with 4 cores, and 16 GB (GigaBytes) of RAM (Random Access Memory) and an NVIDIA GeForce GTX 1070. Only SciBERT and SciDr-in used the GPU.

Acronym and In-expansion Technique	SciAI							
	Acronym				Pair			
	P	R	F1	K	P	R	F1	K
SH	96.02%	82.36%	88.67%	0.97	92.85%	79.64%	85.74%	0.88
MadDog-in	**98.63%**	86.72%	92.30%	**0.98**	**96.91%**	85.21%	90.68%	**0.96**
SciBERT	95.69%	**94.05%**	94.86%	0.97	92.21%	**90.64%**	91.42%	0.94
SciBERT with External data	96.18%	**94.05%**	94.90%	0.97	92.50%	90.45%	91.46%	0.93
SciDr-in	97.47%	92.47%	**95.11%**	**0.98**	94.47%	89.63%	**91.98%**	0.95
SciDr-in with External data	97.58%	91.78%	94.59%	**0.98**	93.81%	88.24%	90.94%	0.94

Table 4.3: In-expansion techniques Precision, Recall, and F1-measures and Cohen's kappas (K) for acronym and pair extraction and for the SciAI dataset.

Results.

We report Precision, Recall, F1-measure, and Cohen's kappa (K) values for the biomedical datasets

Acronym and In-expansion Technique	User Generated							
	Acronym				Pair			
	P	R	F1	K	P	R	F1	K
SH	91.00%	**70.54%**	79.47%	**0.72**	86.00%	**66.67%**	75.10%	**0.14**
MadDog-in	**92.78%**	69.76%	**79.64%**	0.71	**88.65%**	**66.67%**	**76.10%**	0.11
SciBERT	65.62%	48.83%	55.99%	0.62	58.34%	43.41%	49.77%	0.08
SciBERT with External data	49.67%	58.91%	53.90%	0.62	45.09%	53.48%	48.93%	0.08
SciDr-in	77.08%	57.36%	65.77%	0.66	68.75%	51.16%	58.66%	0.09
SciDr-in with External data	86.36%	58.91%	70.04%	0.69	81.81%	55.81%	66.35%	0.11

Table 4.4: In-expansion techniques Precision, Recall, F1-measures and Cohen's kappas (K) for acronym and pair extraction and for the User Generated dataset.

Acronym and In-expansion Technique	Train models execution times (s)							
	Train Dataset						with External Data	
	Medstract	SH	BIOADI	Ab3P	SciAI	User-Generated	All Biomedical	All
SciBERT	108	745	806	831	1 701	421	2 691	5 122
SciDR-in	1 226	7 255	8 060	8 738	52 257	2 752	28 612	99 025

Table 4.5: In-expansion train models execution times for each dataset and when trained with External Data.

Average execution times per document (s)							
Acronym and In-expansion Technique	Medstract	SH	BIOADI	Ab3P	SciAI	User Generated	Average
SH	**0.00**	**0.02**	**0.01**	**0.01**	**0.05**	**0.00**	**0.02**
MadDog-in	0.60	4.33	7.23	5.04	15.75	4.53	6.25
SciBERT	10.35	70.29	115.23	80.81	686.44	35.97	166.52
SciBERT with External Data	16.80	73.45	122.73	76.80	645.70	42.84	163.05
SciDr-in	165.69	1 039.45	1 831.67	1 225.03	11 105.72	470.88	2 639.74
SciDr-in with External Data	172.81	1 073.44	1 836.02	1 198.92	11 234.76	486.81	2 667.13

Table 4.6: In-expansion average execution times per document for each dataset.

(i.e., Medstract, Schwartz and Hearst, BIOADI and Ab3P) for acronym extraction in Table 4.1 and for acronym-expansion pair extraction in Table 4.2. We report Precision, Recall, F1-measure, and Cohen's kappa (K) values for both acronym and pair for SciAI dataset in Table 4.3 and for End-to-end dataset in Table 4.4. The additional external data used to train SciBERT and SciDr-in for the biomedical application includes the data of all biomedical datasets excluding the test set (30%). For SciAI and End-to-end datasets, the external data used to train SciBERT and SciDr-in includes all documents in the other datasets (i.e., Medstract, Schwartz and Hearst, BIOADI, Ab3p, SciAI, and End-to-end).

For the *biomedical domain*, for all acronym and pair extraction measures (precision, recall, and F1-measure), the best acronym and in-expander technique, on average, is SH.

On *SciAI*, the machine-learning techniques SciDr-in and SciBERT outperform the rule-based techniques, SH and MadDog-in, in terms of recall (**91.78%-94.05%** for acronym and **88.24%-90.64%** for pair) and F1-measure (**94.59%-94.05%** for acronym and **90.94%-91.98%** for pair) for both acronym and pair extraction. Furthermore, MadDog-in achieves the best precisions (**98.63%** for acronym and **96.91%** for pair) followed by SciDr-in (**97.47%-97.58%** for acronym and **93.81%-94.47%** for pair).

On the *End-to-end dataset*, MadDog-in achieves the best overall precision (**92.78%** and **88.65%**) and F1-measure (**79.64%** and **76.10%**) for acronym and pair extraction, while SH surpasses MadDog-in in terms of acronym recall (**70.54%**) and matches for pair recall (**66.67%**). Furthermore, among the machine-learning techniques on the End-to-end dataset, SciDr-in surpasses SciBERT on precision (**77.08%** and **68.75%**), recall (**57.36%** and **51.16%**), and F1-measure (**57.36%** and **58.66%**) for both acronym and pair extraction. Increasing the training dataset with External Data for SciBERT yielded

an increase in recalls (**58.91%** and **53.48%**) but decreased precisions (**49.67%** and **45.09%**) and F1-measures (**53.90%** and **48.93%**) for both acronym and pair extraction, while for SciDr-in, we observe a general increase in performance.

Cohen's Kappa analyzes. In general, the Cohen's Kappa (K) best technique for each dataset is consistent with the F1-measure in the biomedical datasets, i.e., SH technique is clearly the best. On the *SciAI dataset*, MadDog-in scores the best K for acronyms and pairs with **0.98** and **0.96**, respectively.

On the *End-to-end dataset*, all techniques achieve values of Cohen's Kappa higher or equal to **0.62** for acronym extraction and lower or equal to **0.14** for pair extraction. The reason is that the end-to-end data contains many distinct expansions with different frequencies, leading to lower scores.

Glossary-level differences. In Annex B, we report the results obtained using the Glossary-level metrics for each dataset. Regarding the glossary-level results, they are globally very similar to the document-level ones in terms of the differences among in-expansion techniques. The exceptions in the best methods for each metric are small differences: On the BIOADI dataset, the best score for acronym extraction recall is obtained with SciDr-in trained in all biomedical datasets with **79.71%**, SH (which was the best on document-level) scores **78.26%**. On the SciAI dataset, acronym recall is better for SciBERT trained on SciAI training set with **93.58%**, SciBERT trained on all datasets (which was the best on document-level) scored **93.49%**. On the User Generated dataset, the best technique for acronym and the pair extraction F1-measure is SH with **79.49%** and **66.67%** respectively, while Maddog-in (the best on document-level for acronym F1 and one of the best for pair F1) scored **79.04%** for acronym F1, and **65.83%** for pair F1.

Interpretation: In this in-expansion benchmark, rule-based techniques SH and MadDog-in generally perform best for all datasets. The one exception is on the SciAI dataset where machine learning techniques from SciDr-in and SciBERT work better.

Rule-based systems work well for in-expansion, because acronyms follow human-understood rules, viz. roughly, acronyms should be in upper-case, each letter should represent a word, and the expansion should either precede or follow the first use. So it is natural that a rule-based system would do well. Machine learning work better when given more examples (SciAI dataset), however even ensembled with a rule-based technique (SciDr) the results were generally inferior to using the rule-based technique by itself.

While the expansions found by the rule-based techniques are not a superset of those found by the machine learning techniques, SciDr often fails because it adds extra words to the expansion string. On the other hand, SciDr can find unusual cases where not all acronym chars belong in the expansion, e.g., expansion *PIN-FORMED* of *pin1*.

Execution time analysis. We report in Table 4.6 the in-expansion execution times per document. We observed from our experiments that the rule-based techniques are much faster than the machine learning techniques. SH is the fastest technique on every single dataset taking less than **0.06** seconds on average to extract acronym-expansion pairs from a document. MadDog-in is the second fastest technique taking up to **16** seconds to process each document. The machine-learning techniques SciDr-in and SciBERT take much more time to extract pairs from the datasets than the rule-based techniques.

For instance, on the SciAI dataset, SciBERT takes **687** seconds. Comparing SciDr-in and SciBERT, the execution times per document of SciDr-in are much higher than SciBERT taking **11 106** seconds on the SciAI dataset because it is an ensemble based technique. We report in Table 4.5 the train execution times for machine-learning based techniques (SciBERT and SciDr-in) for each dataset and with External Data. Regarding training times, SciBERT takes **1 700** seconds on SciAI training data, **2 691** seconds on the biomedical datasets, and **5 121** seconds to train with all datasets. SciDr-in takes longer for training: **52 257** seconds on SciAI training data, **28 612** seconds on the biomedical datasets, and **99 024** seconds to train with all datasets.

In summary:

- If document processing needs to be fast and there are hardware limitations, the rule-based techniques **SH** and **MadDog-in** are the best.
- If there are no time constraints and a large volume of data from the same domain is available, the machine learning techniques **SciBert** and **SciDr-in** are marginally better.
- If the dataset is sentence-based and a large amount of data from the same domain is available, use **SciDr-in**, though, in the medical domain, **SH** would likely be a strong contender.

4.3 Out-expansion Benchmark, Evaluation and Results

We describe our benchmark of out-expansion techniques in Section 4.3.1 and evaluate state-of-the-art techniques on this benchmark in Section 4.3.2.

4.3.1 A Benchmark of Out-expansion Techniques

This section presents our benchmark of out-expansion techniques. Section 4.3.1.1 describes the datasets used in this benchmark. Section 4.3.1.2 explains the steps used to prepare those datasets. Section 4.3.1.3 lists the out-expansion techniques included in the benchmark, grouped by type. Finally, Section 4.3.1.4 describes the metrics to evaluate those out-expansion techniques.

4.3.1.1 Datasets

The datasets included in our out-expansion benchmark are:

MSH dataset [52] contains biomedical document abstracts from the MEDLINE (Medical Literature Analysis and Retrieval System Online) corpus used in Li et al. [61], Prokofyev et al. [85]. This dataset was automatically annotated using citations from MEDLINE and the ambiguous terms with Medical Subject Headings (MSH) identified in the Metathesaurus[10]. We use the original texts and the revised labels from Li et al. [61];

[10] https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus

Statistics	SciWISE	MSH	CS Wiki	SciAD
# of articles	4 677	12 053	10 220	39 815
Average # of chars per article	1 193	1 552	9 246	187
# of sentences	40 300	112 456	702 387	51 340
# of distinct acronyms	129	67	630	732
# of distinct ambiguous acronyms	100	64	566	682
Average # of distinct expansions per article	1.09	0.99	1.02	1
# of distinct acronym/expansions	272	139	8 617	2173
Average # of expansions per ambiguous acronym	2.43	2.13	15.11	3.11

Table 4.7: Statistics of the out-expansion datasets. SciAD is the dataset with the largest number of articles, 39k. However, if we compare the number of sentences found in each dataset, CS Wiki has the largest total with 70K, 11k for MSH, 5k for SciAD and 4k for ScienceWISE. The reason is that SciAD contains the smallest articles in terms of characters (mostly just sentences having fewer than 200 characters), while others have 1k or more. CS Wiki has the biggest set of distinct acronym-expansions pairs (8k). In terms of distinct expansions per ambiguous acronym, CS Wiki has around 15, while the remaining dataset have maximum around 3.

SciWISE dataset consists on the Physics dataset used in Li et al. [61] and Prokofyev et al. [85] that consists of document abstracts. This dataset was annotated by human experts, and it includes expansions either containing at least 2 words or a single word with at least 14 characters.

CS Wiki (Computer Science Wikipedia) dataset created in Thakker et al. [107] contains documents from different fields that contain acronyms used in computer science. Expansions were extracted by parsing the content of English Wikipedia disambiguation pages of acronyms used in computer science (e.g., `https://en.wikipedia.org/wiki/PDF_(disambiguation)`).

SciAD This dataset was prepared for the out-expansion SDU@ AAAI-21 competition [111]. It is based on the SciAI in-expansion dataset, described in Section 4.2.1.1. We use the revised version[11] created by Egan and Bohannon [30] who removed duplicate sentences from the original trainning and validation sets.

Table 4.7 presents relevant statistics about each dataset. An ambiguous acronym is an acronym having more than one expansion available in the dataset.

4.3.1.2 Data Preparation

The data preparation steps are roughly the same for each out-expansion technique:

1. **Dataset Splitting:** We split each dataset into *train* and *test* sets (respectively 70% and 30% of the documents of the original dataset). We then apply 5-fold cross validation on the train dataset in order to tune the hyperparameters of each out-expansion technique. The hyperparameter-tuned technique is then tested on the yet unseen 30% of the data.

2. **Expansion Consolidation:** For the expansions of acronym A in each dataset, we apply an approximate duplicate detection process that groups expansion strings that correspond to the same expansion meaning. For example, *portable document format* and *Portable-Document-Formats* are two distinct strings that refer to the same real expansion. As criteria, we consider two expansions

[11] https://github.com/PrimerAI/sdu-data

to be equal if their lower case versions without dashes have an edit-distance less than 3 or if the first 4 characters of each word are equal. Equal expansions are consolidated by replacing in text all expansion strings with the same meaning by the most frequent expansion.

3. **Expansion Removal:** When testing the accuracy of out-expansion techniques on some document d, we associate any acronym A in the document with its in-expansion $In(A)$, if present. Then, we replace all occurrences of the in-expansion $In(A)$ in text by A alone.

4. **Tokenization:** We apply the word tokenization from the Natural Language Toolkit (NLTK) [11] to obtain only alphanumeric tokens. Additionally, we remove stop words using NLTK and numeric tokens;

5. **Token Normalization:** We transform each token into its stem, e.g., probable, probability, and probabilities all map to probabl. We use the Porter Stemmer algorithm from NLTK.

The preparation of the MSH and SciWISE datasets follows the preprocessing reported in Li et al. [61], so we apply all the preparation steps above except token normalization. The five steps are consistent with the pre-processing steps used in Thakker et al. [107] for the CS Wiki dataset. For SciDr-out and MadDog-out, we apply only the first three steps, because these techniques replace the last two steps with steps that depend on the language models of the neural networks they use.

4.3.1.3 Out-expansion Techniques

This benchmark includes the following groups of out-expansion techniques:

Classical Techniques: We use two baselines: **Random** which randomly assigns a possible expansion to an acronym; and **Most Frequent** which always selects the most frequent expansion found in our training data as measured by the number of occurrences in distinct documents. We use the Cosine similarity (**Cossim**) with the Classic Context Vector (**CCV**) [61], Document Context Vector (**DCV**) - variant of Classic for each document, Surrounding Based Embedding (**SBE**) [61], and **Thakker** et al. [107].

Sentence-oriented Techniques: We include related work techniques that expect a sentence as input (instead of a document) and adapt them as described in the AcX overview (Section 4.1.3). These include Unsupervised Abbreviation Disambiguation (**UAD**) [21], MadDog [110] out-expander (**MadDog-out**), and SciDr [99] out-expander (**SciDr-out**). We also use **SciDr-out with External Data** consisting of the Wikipedia pages that contain an expansion found in the training data.

Representator Techniques: We include **Cossim** with the document representation techniques described in Section 4.1.2, that we have adapted from natural language processing: Term Frequency-Inverse Document Frequency (**TF-IDF**), Latent Dirichlet Allocation (**LDA**), **Doc2Vec**, and Sentence Bidirectional Encoder Representations from Transformers (**SBERT**). We used SBERT model *all-mpnet-base-v2*, the top performing model in Sentence Similarity tasks (14 datasets)[12]. all-pnet-

[12] https://www.sbert.net/docs/pretrained_models.html#model-overview

base-v2[13] is based on MPNet model [101] that outperforms BERT and RoBERTA in both quality and speed. *all-mpnet-base-v2* was trained on one billion sentences pairs from a diverse set of data sources.

Classification Techniques: We created a complete new class of out-expansion techniques that use the outputs of a representator as features for a Machine Learning classifier, specifically, Random Forests (**RF**), Logistic Regression (**LR**), and Support Vector Machines (**SVM**). Each acronym has its own classifier trained with the features of the documents that contain an expansion for the acronym (e.g., acronym PDF will have a random forest RandFor(PDF) based on documents that contain an in-expansion for PDF). Based on the features of a target document d, the classifier will choose the appropriate expansion as explained in Section 4.1.3.

Combination of Representator Techniques: The final type of out-expansion techniques that we assembled consists of combining two representators' outputs, namely the Doc2Vec with a Context Vector (either Classic or Document), as input to predictors: **CCV + Doc2Vec** and **DCV + Doc2Vec**. Combinations are constructed by concatenating the outputs together into a single feature vector.

Ensembler Techniques: We support two ensembler techniques: **Hard** voting where each technique votes for its preferred expansion regardless of its confidence; and **Soft** voting that takes the averages of confidences per expansion. The confidences are normalized at the individual technique level in such a way that their sum is 1. For the experiments, we assembled the following 7 out-expansion techniques: Cossim with CCV, Cossim with TF-IDF, Cossim with Do2Vec, SVM with Doc2Vec, Cossim with SBERT, SVM with SBERT, and SciDr-out.

4.3.1.4 Performance Metrics

Our benchmark uses the following metrics:

Out-expansion accuracy: is the accuracy of predicting the right expansion for a given acronym in a textual document. Intuitively, this is the fraction of acronym-expansions that are correctly predicted. This corresponds to a micro-average of Precision and Recall, but, since we always predict an expansion for some acronym in the out-expansion task, those two metrics are equal. Accuracy is also used in previous out-expansion works [21, 61, 107] and analogous benchmarks, e.g., for Word-Sense-Disambiguation [88]. Note that an acronym may appear many times in the same document and many times across documents. In our measure, if A is in k documents, it is counted k times, but if A is present j times in the same document, it is counted only once in that document. Thus, for a test set of documents D, the out-expansion accuracy is: $\frac{\sum_{d \in D} |\text{correct distinct expansions for d}|}{\sum_{d \in D} |\text{distinct acronyms inside d}|}$.

Out-Expansion macro averages: Recently, Veyseh et al. [110][84] started using a different set of metrics that we have implemented and measured for completeness. Those metrics are macro-averages of Precision, Recall and F1-measures for acronym-expansions pairs. So, we calculate

[13] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

precision, recall, and F1-measure independently for each acronym-expansion in the training data. Then, averages of those measures are performed in order to obtain the final macro averages. Note that, when using these measures, very rare acronyms and expansions in the dataset will have the same impact as more frequent expansions and acronyms.

Representator execution time: is the execution time to create representations of training documents.

Average execution time per document: is the average execution time to predict expansions for acronyms in a document.

4.3.2 Out-expansion Experimental Results

Setup. For out-expansion on the benchmark presented in Section 4.3.1, we ran the experiments on a GoogleCloud platform[14] machine with the following specifications: Intel Broadwell CPU platform with 8 cores, 30GB to 80GB of RAM (Random Access Memory). For SBERT, MadDog-out, SciDr-out, and LUKE half of a Tesla K80 GPU board was used. The code ran in Python 3.7.

To reduce the duration of experiments, we first find the representator's hyperparameters with the cosine similarity because it involves no learning nor hyperparameters of its own, then save the best representator model on disk. Next, given the best representator hyperparmeters, we find the best out-expansion predictor model hyperparameters.

Results. For each dataset, in Table 4.8, we report the out-expansion accuracy and macro F1-measure to predict the expansions of acronyms in a document. In Table 4.9, we present the average execution times that out-expansion techniques that to process a document for each dataset. The *Technique Group* column identifies the out-expansion group that the technique belongs to, as organized in Section 4.3.1.3 (e.g., Classical). The *Predictors* column identifies the out-expansion predictor technique (e.g., Cossim or an ML classifier) that takes a given document representation to predict an expansion (e.g., Cossim). The *Representators* column indicates the technique used to generate a document representation (e.g., Doc2Vec). We did not run SciDr-out with External Data on CSWiki dataset because the external data (i.e., Wikipedia data) would overlap with CSWiki itself. The execution time of each ensemble technique is just the additional time required to decide on an expansion given the input predictions and confidence measures.

In these out-expansion experiments, we measure the accuracy and macro F1 only on the acronym-expansions pairs whose acronym is ambiguous (i.e., have at least two expansions in the training data) and whose in-expansions are in the training data.

The best individual techniques (average above 89% of accuracy) in descending order are: Cossim with SBERT, SVM with SBERT, SciDr-out, Cossim with CCV, Cossim with TF-IDF, Cossim with DCV, Cossim with Doc2Vec alone or with DCV, and SVM with Doc2Vec. Regarding statistical significance, Cossim with SBERT is the best for SciWISE. For MSH, SVM with Doc2vec combined with either CCV or DCV score higher accuracy. However, they are not statistically significantly better than: SVM with either Doc2Vec or SBERT, Cossim with SBERT, and LR with Doc2Vec. SciDr-out achieves higher accuracy for

[14] `https://cloud.google.com/`

Out-expansion Technique			ScienceWISE		MSH		CSWiki		SciAD		Average	
Technique Group	**Predictors**	**Repre-sentators**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**
Classical	Random		47.72%	46.10%	47.04%	45.49%	14.54%	14.21%	33.06%	32.22%	35.59%	34.51%
	Most Frequent		70.52%	49.31%	50.30%	32.32%	47.76%	20.37%	69.08%	37.64%	59.41%	34.91%
	Cossim	CCV	91.34%	80.72%	97.62%	97.65%	77.96%	65.01%	92.28%	89.03%	89.80%	83.10%
		DCV	89.51%	78.69%	96.18%	96.07%	78.59%	65.86%	93.67%	87.08%	89.49%	81.92%
		SBE	88.07%	75.89%	95.84%	95.24%	74.60%	63.30%	86.50%	80.76%	86.25%	78.80%
	Thakker		87.77%	77.38%	92.53%	91.68%	73.16%	63.86%	84.36%	73.21%	84.46%	76.53%
Entity Disam.	LUKE		83.42%	57.33%	67.47%	58.95%	52.65%	46.60%	50.68%	42.53%	63.55%	51.35%
Sentence--Oriented	UAD		43.69%	46.73%	93.92%	92.55%	12.94%	11.60%	34.98%	45.75%	46.38%	49.16%
	MadDog-out		89.13%	68.84%	94.09%	93.16%	57.03%	47.71%	87.38%	73.23%	81.91%	70.73%
	SciDr-out		88.22%	77.45%	97.23%	96.76%	**84.19%**	**72.67%**	**94.48%**	88.94%	91.03%	83.96%
	SciDr-out with External Data		89.89%	77.86%	97.58%	97.22%	N/A	N/A	**94.71%**	**89.42%**	N/A	N/A
Repre-sentator	Cossim	TF-IDF	91.26%	81.82%	97.62%	97.57%	77.80%	65.36%	91.79%	83.48%	89.62%	82.06%
		LDA	85.56%	73.94%	93.81%	93.28%	71.89%	60.49%	84.56%	73.39%	83.95%	75.28%
		Doc2Vec	92.86%	83.14%	98.33%	98.07%	77.16%	65.25%	92.05%	82.96%	90.10%	82.35%
		SBERT	**94.83%**	**85.32%**	**98.78%**	**98.80%**	81.47%	67.67%	**94.19%**	89.76%	**92.32%**	**85.39%**
Classi-fication	RF	TFIDF	70.82%	52.03%	84.53%	76.78%	32.11%	23.14%	87.64%	68.90%	68.77%	55.21%
		LDA	70.75%	54.13%	95.64%	92.84%	67.57%	50.78%	82.32%	61.33%	79.07%	64.77%
		Doc2Vec	79.18%	61.34%	96.58%	95.37%	66.29%	41.55%	84.39%	62.43%	81.61%	65.17%
	LR	TFIDF	71.05%	54.47%	93.41%	88.29%	71.89%	45.59%	80.63%	55.06%	79.24%	60.85%
		LDA	71.13%	51.49%	88.66%	80.02%	71.73%	48.77%	80.08%	55.16%	77.90%	58.86%
		Doc2Vec	88.83%	78.35%	**98.87%**	98.72%	76.68%	57.97%	90.75%	77.95%	88.78%	78.25%
	SVM	TFIDF	81.84%	62.13%	94.71%	91.27%	77.16%	53.54%	91.01%	78.24%	86.18%	71.29%
		LDA	78.88%	59.80%	93.64%	91.16%	71.89%	51.11%	85.59%	70.63%	82.50%	68.18%
		Doc2Vec	89.67%	79.31%	**98.93%**	98.79%	77.00%	58.70%	91.56%	80.88%	89.29%	79.42%
		SBERT	93.01%	83.91%	**98.87%**	**98.84%**	**82.43%**	64.44%	92.34%	86.53%	91.66%	83.43%
Combi-nation of Repre-sentators	Cossim	CCV + Doc2Vec	90.27%	79.04%	98.19%	97.95%	77.16%	65.25%	86.92%	82.82%	88.14%	81.27%
		DCV + Doc2Vec	90.27%	79.01%	98.33%	98.10%	77.16%	65.19%	92.05%	82.96%	89.45%	81.32%
	SVM	CCV + Doc2Vec	89.97%	80.44%	**98.95%**	**98.84%**	77.00%	58.70%	80.73%	76.06%	86.66%	78.51%
		DCV + Doc2Vec	89.67%	79.35%	**98.95%**	98.83%	77.00%	58.70%	90.20%	75.90%	88.95%	78.20%
Ensemblers	Hard		**94.15%**	**86.95%**	**99.60%**	**99.58%**	**84.19%**	78.32%	**96.59%**	**91.88%**	93.63%	**89.18%**
	Soft		93.62%	85.00%	99.38%	99.41%	**86.26%**	**79.33%**	95.94%	91.04%	**93.80%**	88.70%

Table 4.8: Out-expansion accuracy (Acc) and macro F1-measure (MaF1). Values marked as bold indicate the best Acc obtained by an individual technique and by an ensembler, respectively in that dataset. A technique T1 is considered better than T2 if a non-parametric significance test (based on shuffling[43]) indicates that the difference in their means has a p-value ¡ 0.05. Thus, even though each column has a highest mean value for some technique H which will be bolded, the value of a technique T will also be bolded if H is no better than T based on the p-value criterion. We apply the same p-value criteria to bold ensemblers on all datasets, except on ScienceWISE where we apply the statistical test to each ensembler against Cossim SBERT (the best technique on ScienceWISE).

Out-expansion Technique			Execution Times per Document (s)				
Technique Group	**Predictors**	**Representators**	**SciWISE**	**MSH**	**CSWiki**	**SciAD**	**Average**
Classical	Random		**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Most Frequent		**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Cossim	CCV	0.01	0.06	0.07	0.12	0.07
		DCV	0.02	0.18	0.08	0.34	0.15
		SBE	0.02	0.07	0.06	0.06	0.05
	Thakker		2.19	7.56	2.62	2.78	3.79
Entity Disam.	LUKE		0.88	3.38	63.74	0.31	17.07
Sentence-Oriented	UAD		0.00	0.01	0.01	0.00	0.01
	MadDog-out		0.07	0.22	1.15	0.04	0.37
	SciDr-out		0.72	1.19	2.51	0.70	1.28
	SciDr-out with External Data		2.15	3.37	N/A	1.28	N/A
Representator	Cossim	TF-IDF	0.06	5.02	4.20	0.87	2.53
		LDA	0.01	0.02	0.03	0.02	0.02
		Doc2Vec	0.01	0.01	0.36	0.02	0.10
		SBERT	0.19	0.39	0.50	0.13	0.30
Classification	RF	TFIDF	1.03	17.77	72.85	3.52	23.79
		LDA	1.01	1.45	1.02	1.32	1.20
		Doc2Vec	0.12	1.44	2.14	1.75	1.36
	LR	TFIDF	0.06	5.03	41.36	0.92	11.84
		LDA	0.01	0.02	0.03	0.02	0.02
		Doc2Vec	0.01	0.02	0.36	0.04	0.11
	SVM	TFIDF	0.04	4.77	7.05	1.05	3.23
		LDA	0.01	0.03	0.03	0.02	0.02
		Doc2Vec	0.01	0.01	0.37	0.02	0.10
		SBERT	0.14	0.26	0.59	0.17	0.29
Combination of Representators	Cossim	CCV + Doc2Vec	0.03	0.23	0.65	0.41	0.33
		DCV + Doc2Vec	0.44	2.32	1.29	2.54	1.65
	SVM	CCV + Doc2Vec	0.04	0.29	0.78	0.46	0.39
		DCV + Doc2Vec	0.47	2.36	1.37	2.51	1.68
Ensemblers	Hard		0.00	0.00	0.00	0.00	0.00
	Soft		0.00	0.00	0.00	0.00	0.00

Table 4.9: Overall out-expansion techniques average execution times per document.

CSWiki, but is not statistically better than SVM with SBERT. Finally, for SciAD, SciDr-out with external data scores higher accuracy but not statistically significantly better than: SciDr-out and Cossim with SBERT.

Interpretation: An important question in interpreting these numerical results is to understand why some techniques are better than others.

For out-expansion, the best approaches SciDr-out and Cossim/ SVM with SBERT are based on language models trained on large data collections, but that does not tell the whole story. SciDr-out uses the particularly effective strategy of predicting the expansion span from the list of possible expansions passed as input. Further, SciDr-out is an ensemble of models trained in a 5-fold cross-validation setting. SBERT augments transformer language models to sentence similarity tasks using a siamese

Representators	**SciWISE**	**MSH**	**CSWiki**	**SciAD**	**Average**
CCV	**0**	**1**	**5**	**1**	**2**
DCV	**0**	**1**	**5**	**1**	**2**
SBE	6	23	115	9	38
LUKE	1 917	9 448	29 319	9 435	12 529
UAD	43	93	331	54	130
TF-IDF	2	10	58	1	18
LDA	155	7 766	8 830	4 247	5 250
Doc2Vec	13	32	212	108	91
SBERT	111	437	1 860	280	672
SciDr-in	11 227	42 716	147 454	50 282	62 920
SciDr-in External Data	14 299	46 651	N/A	66 441	42 464
MadDog-out	566	728	10 008	1 198	3 125

Table 4.10: Representator execution times in seconds for each dataset.

Out-expansion Technique		SciWISE	MSH	CSWiki	SciAD	Average
Predictors	**Representators**					
Cossim	CCV	0.35	0.25	0.38	0.34	0.33
	TF-IDF	0.37	0.33	0.48	0.30	0.37
	Doc2Vec	0.27	0.14	0.32	0.19	0.23
	SBERT	0.24	0.15	0.32	0.24	0.24
SVM	Doc2Vec	0.10	0.03	0.36	0.19	0.17
	SBERT	0.07	0.06	0.34	0.23	0.18

Table 4.11: Pearson correlation coefficient values of confidence with correct acronym expansion for each dataset and each technique. Accuracy is correlated with confidence, but only modestly.

architecture that generates embeddings for each sentence and is trained to maximize similarity. Those embeddings turn out to be very informative regarding the context for documents: both Cossim or SVM combined with SBERT obtained on average the highest accuracy among individual techniques.

While LUKE's transformer language model enables the creation of entity embeddings (in LUKE's case, fine-tuned for expansion embeddings), the results are not the best for acronyms, even with fine-tuning. One reason is that each entity is referenced frequently (over 600 times on the average [115]). Acronym/expansion pairs are referenced less than twice on the average. For example, Wikipedia has 11 million entity occurrences and 18 thousand distinct entities [115], an average of 611 mentions per entity. By contrast, examples of sentences whose acronyms are expanded by links are limited, because some acronyms may be defined in a single document. On CSWiki, for example, we have 10,400 acronym expansion occurrences (without counting repeated occurrences in the same document) and 8,600 distinct acronym/expansion pairs, an average of only 1.2 occurrences per acronym expansion.

Independently of which technique is best, we should note that each of the top techniques, except SciDr-out, gives a confidence score. For some of the best techniques SBERT, Doc2Vec, TFIDF, and CCV, the confidence score has a positive correlation with accuracy, though the correlation is modest (under 0.5). as indicated in Table 4.11. This low positive correlation is reflected in our results for ensemble techniques. The soft ensemble technique (in which each underlying technique's weight is monotonic with its confidence) does well thanks to the positive correlation. On the other hand, hard voting ensemble techniques (in which each underlying technique votes for its preferred expansion regardless of confidence) perform even better, suggesting that the "wisdom of crowds" effect is stronger than using confidences. A deeper look at ensemble techniques for acronym expansion is a subject for future work.

Analysis of Classical techniques. The baselines Random and Most frequent out-expanders have better accuracies on SciWISE (**48%** and **71%**) followed by SciAD (**33%** and **69%**) and MSH (**47%** and **50%**). The worst scores are obtained on CSWiki (**15%** and **48%**), because that dataset has far more ambiguity.

MadDog-out achieves near the best performance on SciWISE and MSH, is also good on SciAD, and is worst on CSWiki. MadDog-out's input is a sentence at a time like SBE and UAD, so context is also limited to a few words. In contrast, the best techniques, except SciDr-out, process all words in a document.

Analysis of remaining Representators and Classification techniques. LDA works well in many Natural Language Processing tasks, but less well for our out-expansion task, probably because measuring document similarity by comparing topics is not the best use of LDA.

Techniques with TF-IDF as a representator are surprisingly competitive in SciWISE and MSH, but they achieve inferior accuracy in CSWiki. Pre-processing steps (Section 4.3.1.2) play an important role in CCV performance, for instance, in the Tokenization step, we remove stop-words which TF-IDF takes into account thus automatically giving them less relevance due to the IDF score.

Predictors using RF are slower and slightly less accurate than Cossim and SVMs. RFs struggle with the fact that, in this acronym expansion, there are many features but only a small number of samples, e.g., 300 dimensions from Doc2Vec and a few documents per acronym. By contrast, Cossim and SVM are usually the best predictors closely followed by LR. The representator hyperparameter search was performed for TF-IDF, LDA, and Doc2Vec using Cossim as a predictor, hence they are fitted for this predictor. SVMs and LRs are similar classifiers, the first fits a separating hyper-plane, the other a logistic function.

Execution times of representators. The training execution times depend only on the representators and are reported in Table 4.10. The Doc2Vec models used by Thakker et al. are created during the input document processing hence are not reported in Table 4.10.

The CCV and DCV representators take the least time (average **2s**) closely followed by TF-IDF (average **18s**) (Table 4.10). By contrast, Word2Vec (SBE and UAD) and Doc2Vec models take more time depending on the hyperparameters and dataset size (**6-331s**). LDA takes on average **5ks**. The most expensive models are SciDr-out (**14ks-66ks**) followed by LUKE (**1Ks-13ks**) and MadDog-out (**566s-10ks**) which use either language models or neural networks. Thakker et al. [107] does not report execution time to produce the representator model, because the system builds a Doc2Vec per acronym during the input document processing time.

Document processing execution times. As observed in Table4.9, among these best techniques, Cossim with CCV is the fastest for all datasets, able to process input documents in less than **0.07** seconds on dataset average. However, SVM with Doc2Vec is the fastest for MSH and SciWISE. The slowest among the best is Cossim with TF-IDF (average **2.5s**), followed by SciDr-out (**1.3s** for base and **2.3s** with external data). These differences are statistically significant.

External data analysis. SciDr-out with External Data improves over the SciDr-out base (Table 4.8), indicating that adding additional models trained on external data usually helps at roughly double the time cost. The downside is that training time more than doubles (we have to sum the SciDr-out External data in Table 4.10). Average execution times (Table 4.8) to process a document almost doubles as well (from **1.3s** to **2.3s**).

In summary:

- If neither training time nor document processing time is of major concern and especially if GPU processing is available, then use either a **Hard** ensembler (best but slowest), **SciDr-out** (best with more domain data) or **Cossim**/**SVM** with **SBERT** (fastest and close to best).
- A pipeline that balances time and accuracy is to use **Doc2vec** as feature inputs for either **Cossim** or **SVMs**.
- If training and test time is limited, use **Cossim** with **CCV**, which requires almost no training time

(less than 5s) and is the fastest in testing time among the best set of techniques.

4.4 End-to-end Benchmark and Evaluation

The end-to-end benchmark described in Section 4.4.1 uses input documents containing acronyms, where some of those acronyms contain expansions and others do not. The evaluation in Section 4.4.2 will measure the recall and precision of acronym expansion for both student annotators and AcX pipelines.

4.4.1 A Benchmark of End-to-End Acronym Expansion

Section 4.4.1.1 describes the train and test datasets used in this benchmark. Section 4.4.1.2 lists the end-to-end acronym expander systems included in the benchmark. Finally, Section 4.4.1.3 presents the metrics that our benchmark uses to evaluate the systems.

4.4.1.1 Datasets

The end-to-end benchmark uses two different datasets: (i) for testing, the end-to-end dataset of Section 4.2.1.1. (ii) The *train* dataset consists of documents from Wikipedia that do not belong to the annotated test set. Those documents came from the Wikipedia dump of March 1, 2020[15]. These were converted to pure text using WikiExtractor [6].

We preprocessed all the documents using all the steps described in Section 4.3.1.2 for all out-expansion techniques except MadDog-out which uses its own preprocessing techniques.

4.4.1.2 End-to-end systems

We use: (i) the end-to-end MadDog System (**MadDog-sys**) and (ii) various ***pipelines* of AcX** consisting of an in-expansion technique along with possibly the Link Follower (**LF**) technique followed by an out-expansion technique possibly with machine learning (see Figure 4.1). An example of a pipeline would be the SH in-expander, followed by the LF component, Doc2Vec, and SVMs. The pipelines we test consist of combinations of the most practical (accurate and fastest) techniques for in-expansion and out-expansion as determined by the benchmarks in Sections 4.2.2 and 4.3.2. Specifically, AcX pipelines use either the **MadDog-in** or the **SH** technique as in-expanders to identify acronyms and expansions in input documents. For out-expansion, AcX pipelines include one of the following combinations of out-expansion techniques, i.e., a predictor (Section 4.1.3) with a representator (Section 4.1.2): (i) **Cossim** with **SBERT**; (ii) **SVM** with **SBERT**; (iii) **Cossim** with **CCV**; (iv) **Cossim** with **Doc2vec**; and (v) **SVM** with **Doc2vec**.

Finally, we compare the accuracies of MadDog-sys, the various pipelines of AcX, and the **student annotators** (before the reviewer, the third annotator, resolved conflicts to decide on the final annotations).

[15] `https://dumps.wikimedia.org/enwiki`

AcX (pipelines)				Out-expansion			End-to-end			Execution Time per Document (s)
In-expander	Out-expander Predictor	Representator	LF	P	R	F1	P	R	F1	
SH	Cossim	CCV	No	44.00%	37.40%	40.44%	51.43%	45.62%	48.35%	21.31
			Yes	34.66%	22.68%	27.42%	52.16%	46.30%	49.05%	18.51
		SBERT	No	46.03%	39.12%	42.30%	53.10%	47.10%	49.92%	**0.15**
			Yes	36.70%	23.99%	29.01%	53.34%	47.35%	50.16%	1.16
	SVM	Doc2Vec	Yes	40.86%	26.74%	32.33%	56.05%	49.75%	52.71%	3.26
		SBERT	No	51.10%	**43.43%**	46.95%	57.27%	**50.80%**	**53.84%**	2.37
			Yes	42.02%	27.46%	33.22%	56.68%	**50.31%**	53.30%	2.91
MadDog-in	Cossim	CCV	No	44.77%	34.62%	39.05%	53.12%	43.15%	47.62%	17.45
			Yes	37.36%	21.85%	27.57%	54.46%	44.44%	48.95%	19.51
		Doc2Vec	Yes	40.20%	23.50%	29.66%	56.20%	45.86%	50.51%	7.73
		SBERT	No	**57.27%**	36.55%	41.23%	55.17%	44.81%	49.46%	0.33
			Yes	40.02%	23.29%	29.44%	56.28%	45.93%	50.58%	7.76
	SVM	Doc2Vec	No	52.08%	40.27%	45.42%	59.12%	48.02%	53.00%	1.12
			Yes	45.38%	26.53%	33.49%	59.38%	48.46%	53.37%	7.63
		SBERT	No	54.76%	42.35%	**47.76%**	**61.32%**	**49.81%**	**54.97%**	2.29
			Yes	47.09%	27.40%	34.64%	60.59%	**49.44%**	**54.45%**	9.12
MadDog-sys				25.40%	18.38%	21.33%	37.85%	29.14%	32.93%	1084.92
Student annotators				N/A	N/A	N/A	88.36%	76.41%	81.95%	N/A

Table 4.12: Out-expansion and end-to-end system quality metrics and average execution times to process a document in seconds. The in-expander technique and the Link Follower (LF) component provide the input for out-expansion techniques, so when those two components are not fixed, we cannot compare out-expansion techniques directly using the out-expansion quality metrics. End-to-end values marked as bold indicate the best obtained in that metric. A method M1 is considered better than M2 if a non-parametric significance test (based on shuffling[43]) indicates that the difference in their means has a p-value ¡ 0.05. Thus, even though each column has a highest mean value for some method H, the value of a method M will be bolded if H is no better than M based on the p-value criterion.

In-expander Technique	In-expansion			LF
	P	R	F1	P
SH	86.17%	**57.37%**	68.88%	**77.66%**
MadDog-in	**91.49%**	56.58%	**69.92%**	73.19%

Table 4.13: In-expansion prediction, recall, and F1-measure and Link Follower (LF) prediction when each in-expander technique is used.

4.4.1.3 Performance Metrics

Similarly to Section 4.2.1.3, we evaluate MadDog-sys, different pipelines of AcX, and human annotators listed in Section 4.4.1.2 in terms of Precision (**P**), Recall (**R**) and F1-Measure (**F1**). *Precision* is the number of correct system-found acronym-expansion pairs divided by the total number of system-identified pairs. *Recall* is the number of correct system-found acronym-expansion pairs divided by the total number of human-found pairs. *F1-measure* is the harmonic mean of Precision and Recall. We also measure training and per test document execution times.

4.4.2 Results on End-to-end Experiments

Setup. For these experiments, we used a virtual machine with the following specifications: AMD EPYC Processor with 16 cores and 256GB of RAM (Random Access Memory). For SBERT, the virtual machine specifications were: five cores of an Intel Xeon Gold 6126 Processor, 40GB of RAM and a NVIDIA GeForce RTX 2080 Ti. The code ran in Python 3.7.

Results. Table 4.12 presents the results for the AcX system running each one of the different pipelines mentioned in Section 4.4.1.2, the MadDog-sys[16], and the results for the student annotators. More-

[16] `https://archive.org/details/MadDog-models`

Out-Expansion Representator	In-expansion Technique		
	SH	MadDog-in	Average
CCV	**389**	**332**	**361**
Doc2Vec	15,985	13,341	14,663
SBERT	42,769	33,724	38,247

Table 4.14: Representator execution times in seconds for each in-expander technique used to process the train dataset.

over, apart from the end-to-end quality metrics in Table 4.12, we also report the quality metrics for out-expansion (i.e., acronyms left to expand after in-expansion and LF component).

The AcX pipeline composed by MadDog-in, SVM with SBERT without Link Following (LF) obtains the best results with precision (61.32%) and F1-measure (54.97%). However, based on the F1-measure, this is not statistically significantly better (i.e., P-value above 0.05) than the following system pipelines: (i) with the same out-expander but with LF, or (ii) with SH and SVM with SBERT, without LF. The best system pipeline takes **2s** on average to process a document. Our best AcX pipeline obtains better results for all measures than the MadDog-sys (**+20%** of F1) and is faster (**2s** to **1084s**).

Best AcX pipeline analysis. The reason why the precision is low is that our best AcX pipeline considers certain strings to be acronyms even though they are not (245 in total). Some are small words like ”and” and ”not”. Others are codes like ZAB and ZAU that refer to airports. Conversely, the acronym and in-expansion extraction component fails to identify lower case acronyms as acronyms, common measurement units (e.g., m for meter, g for gram, kbit for kilobit) and some common language abbreviations (e.g., Micro, ”etc”, email) which usually everyone knows. By contrast, AcX provides the correct expansion for the acronyms that newcomers to a field may not know, e.g., CAS - Computer Algebra System; SLS - SoftLanding Linux system; and ILM - Industrial Light & Magic.

In and out expansion analysis. We report independently the performance of in-expansion in Table 4.13 and out-expansion in Table 4.12. When evaluating just the acronym and in-expander extraction component of AcX pipelines, using SH scored an in-expansion F1-measure of **68.88%** and using MadDog-in scored **69.92%**. If we evaluate out-expansion (acronyms left to expand after in-expansion and LF component), our best AcX pipeline (SVM as predictor with SBERT as representator) obtains an F1-measure of **47.76%**.

LF analysis. Overall, the Link Following component improves the quality metrics marginally if at all for these link-poor datasets. Unsurprisingly, the better the out-expander, the less LF helps.

We report Precision for only the LF predictions in Table 4.13: **77.66%** when running with SH and **73.19%** when running with MadDog-in. When combined with the best out-expander techniques (i.e., SVM as predictor with SBERT as representator), following links reduces the score by **-0.52%**, however when we consider the second-best set of out-expander techniques (i.e., SVM as predictor with Doc2Vec as representator), LF improves the F1-score by **+0.37%** (Table 4.12). Thus, the better the out-expander, the less link following helps.

For the link follower, execution times were not measured for the time to download a linked document. When we predict an acronym with LF then it is an acronym less to out-expand. For SVM with SBERT or Doc2Vec we have increments of **6-7s** by following links with MadDog-in, however for Cossim with CCV

with MadDog-in0 and follow links with SH we observe a small difference. If the out-expansion technique is slow, then following links may also improve execution times.

Execution times of representators. We report in Table 4.14, the execution times to create the out-expansion representators for each system pipeline. We observe that CCV executes in less time (**389s** with SH and **332s** with MadDog-in) than Doc2Vec (**15,985s** with SH and **,13 341s** with MadDog-in). SBERT representator takes the longest time (**42,769** with SH and **33,724s** with MadDog-in)

Comparison with human performance. Compared with human annotators, our best AcX pipeline (MadDog-in and SVMs with SBERT) is around 27% lower in Precision, Recall, and F1-Measure. So, there is a lot of room for improvement. On the other hand, automatic Acronym Expansion is rapid (**2s** per document) and can give at least a good first guess.

An example application of AcX. Consider one of the documents out of the 163 at random whose original page is here `https://en.wikipedia.org/wiki/CC_(complexity)`. Our best AcX pipeline identified the following acronym-expansion pairs: CC - comparator circuits; CCVP - comparator circuit value problem; AC - alternating current; NC - nick's class; and NL - national league. However, it failed to identify CC-complete, and P. We can see that CC, CCVP, NC, and AL are correct and NL is incorrect. With a different Pipeline consisting of Doc2Vec instead of SBERT, AL is incorrect, but NL is correct.

In summary:

- The best AcX pipeline consists of **MadDog-in**, with **SVM** and **SBERT**.
- The **Link Following** component has high precision but does not improve AcX performance compared with the best pipeline, though that could change depending on the density of acronym-related links.

Chapter 5

Support for User Involvement

In this chapter, we report the work performed to improve the support for user involvement when executing a data cleaning process. This work was a collaboration with Professors Antónia Lopes and Manuel J. Fonseca, both affiliated to *LaSIGE, Faculdade de Ciências, Universidade de Lisboa*.

User involvement is required during the execution of a data cleaning process because: (i) data transformations used in a data cleaning process need to be tuned, and (ii) data records that are generated by the transformations that compose a data cleaning process may need to be manually repaired through user feedback during the execution of an automatic data cleaning process.

Similarly to software development, a data cleaning process needs to be iteratively refined and executed to obtain the highest quality of data. These refinements can be due to: (i) new data batches that are continuously provided as input and contain new data quality problems, and (ii) modifications to data transformations that may introduce new data quality problems. Unlike software development, a data cleaning process often benefits from manually data repairs where the user feedback resolves data quality problems, e.g., using expert knowledge to impute missing values. These repairs allow to fix specific data quality problems, that cannot be repaired automatically because they need user expert knowledge. Moreover, if these repairs are introduced in early phases of the data cleaning process, they can avoid the propagation of data quality problems to further stages of the process.

In the literature, [92] describes a design vision for an end-to-end data cleaning framework that includes several types of user involvements during a data cleaning process, such as, refining a data cleaning program and user feedback to manually clean data. The authors of [83] purpose a framework to guide the user for data wrangling tasks including data cleaning, then such tasks are modeled as optimization problems. This framework would perform Human-in-the-loop interaction to select the best data transformations to produce the desired outcome (e.g., clean data). Other research works in data cleaning explore the user involvement in specific data cleaning tasks like: guiding the user to provide manual data repairs that train machine learning models [114], request the user to impute missing values and to validate general rules based on the imputations [44, 90], approximate duplicate detection and elimination [104], or the user involvement in crowdsourcing settings [72, 113]. Other complementary tasks in data quality like data profiling and exploration have benefited from user involvement to detect

data quality problems and efficiently explore data [82, 108].

However, user involvement in ETL (Extract, Transform and Load) tools like Pentaho Data Integration (PDI) [1] is only supported to tune and combine data transformations.

Data wrangling tools like OpenRefine[2] support manual data repairs and automatic data transformation. However, they are limited. In fact, they do not allow modifications to applied manual data repairs and data transformations. The only possibility is to remove manual data repairs or data transformations, but that is also limited by order of newest to oldest in a similar manner to the Undo function present in word processing softwares. So far, commercial data cleaning tools (i.e., ETL and data wrangling tools) do not support the incorporation of user feedback to manually repair data during the iterative refinement and execution of a data cleaning process. Moreover, there is no evaluation of data cleaning tools that compares their efficiency and effectiveness in terms of user involvement.

This chapter describes the following contributions:

- **Cleenex**, a data cleaning framework that is based on the Ajax data cleaning framework [36]. It extends Ajax with the manual data repair component whose main principles were proposed in [37]. This component allows the designer to specify where and how data can be manually repaired during the execution of a data cleaning process. Additionally, Cleenex offers a debugging mechanism [109] for data cleaning processes that provides the provenance of the data. To support the user involvement in an iterative execution of a data cleaning process, we developed the *manual data repairs persistence component* and the *manual data repairs recovery component* that prevent the integral re-execution of data cleaning processes and the loss of previous manual data repairs. Moreover, we formulated and implemented an *algorithm for the detection and automatic recovery of manual data repairs*, that given a new execution of a data cleaning process, identifies which manual data repairs can be automatically re-applied. Even when manual data repairs cannot be automatically re-applied, Cleenex tries to recover those manual data repairs by requesting the user for additional feedback, which is less demanding than manually repairing the data from scratch.

- An **extensive user involvement evaluation of Cleenex** that includes two evaluation studies: (i) the *evaluation of the three individual Cleenex components that support the user involvement* (i.e., manual data repairs, persistence, and recovery); and (ii) the *evaluation of Cleenex against two other data cleaning tools*, OpenRefine and Pentaho Data Integration (PDI), in terms of their efficiency and effectiveness to support the user involvement. To evaluate the Cleenex components, we programmed a simulated and ideal user to execute a data cleaning process, to manually repair the data, and to provide feedback during an iterative data cleaning process. We measured the required amount of data that the simulated user has to visualize and the number of actions that the simulated user has to perform to clean the data.

  To evaluate Cleenex against OpenRefine and PDI, we performed an experimental evaluation with users that include: (i) two datasets with data quality problems that simulate real scenarios, (ii) two tasks in the context of data cleaning that need to be executed by users using each tool, and (iii)

[1] http://community.pentaho.com/projects/data-integration/
[2] https://openrefine.org/

a satisfaction questionnaire that assessed the user perception about each tool. The execution of data cleaning tasks was evaluated by its correct or incorrect completion, execution time and number of user actions (e.g., clicks and keys pressed).

In Section 5.1, we describe Cleenex and the work involved to support the user involvement in an iterative execution of a data cleaning process. In Section 5.2, we describe the experimental evaluation process to evaluate the support for the user involvement in a data cleaning process and the corresponding results.

5.1 Cleenex

In this section, we describe Cleenex[3], a new data cleaning framework that is based on the data cleaning framework called Ajax [36] which separates the logical specification of a data cleaning process from its physical implementation (SQL or Java algorithm implementations). The development of a data cleaning process in Cleenex involves the design of a data cleaning graph where the nodes are data operators or relational tables and the edges connect relational tables to data operators, as input or output. It supports five customizable data operators: (i) Map that supports one-to-many operations that transform a tuple into one or more tuples, usually a split operation; (ii) Match that computes an approximate join between two tables; (iii) Cluster that groups the tuples of a table into partitions, (iv) Merge that receives a table and defines a tuple and its attributes for each partition resulted from some grouping criteria, and (v) View that supports the operations already present in the SQL language. Cleenex also supports a debugging mechanism developed in [109] that enables the user to select tuples from any table in the graph and navigate backwards or forwards through the data cleaning graph displaying the provenance of these tuples.

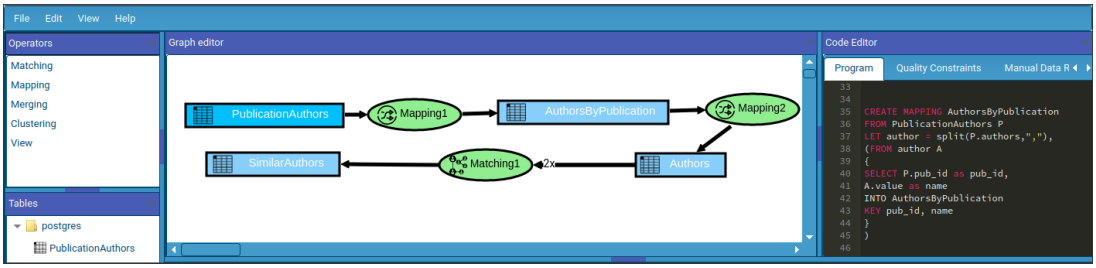


Figure 5.1: Cleenex GUI with a data cleaning graph to detect approximate duplicate authors.

Example 5.1.1. Figure 5.1 presents the Cleenex Graphical User Interface (GUI) together with a data cleaning graph that performs the same data transformations of Example 2.2.1 (Section 2.2.1) to detect approximate duplicate authors from a list of publications (Table *PublicationAuthors*). The Splitting high level data transformation is implemented in Cleenex with two Map operators, the first applies a splitting function for each publication, the second generates Ids for each author name. The Approximate Dupli-

[3]The candidate was part of the developing team of the Cleenex data cleaning framework and responsible for code stability, code reviewing, architectural design, and documentation.

cate Detection is implemented with the Match operator that computes the edit-distance for each author name pair and returns the pairs whose distance value is less than 3.

In order to effectively support the user involvement, the designer may also specify Quality Constraints and/or Manual Data Repairs over any table of a data cleaning graph as proposed in [37]. *Quality Constraints* (QCs) are integrity rules used to assess the data quality in a specific table of the graph. The tuples of a table that violate a quality constraint are named blamed tuples. Those blamed tuples bring the attention of the user to data tuples that do not satisfy QCs. *Manual Data Repairs* (MDRs) allow the user to manually modify data records that belong to tables in a data cleaning graph. In particular, MDRs can be used to manually correct blamed tuples. A MDR is defined by the set of allowed actions that can be taken over a table and a view. The actions can be: (i) updating a specific set of attribute values, (ii) deleting tuples and/or (iii) adding tuples. The view mimics an SQL view and is used to specify the tuples and attributes of a relational table in the graph or the set of blamed tuples that are shown to the user. When the user manually repairs a data tuple, a MDR instance is created. An MDR instance stores the MDR name and the specific manual data modification performed by the user. MDR instances are then used to apply the user manual data repair to the relational table when the data cleaning process re-executes.

Example 5.1.2. User feedback in a data cleaning process. Consider a table *PublicationAuthors* that stores publication ids and its author names as shown in Figure 5.2. A User executes the data cleaning process explained in Example 5.1.1. To simplify, we focus on the first data transformation, where we apply a *Splitting* transformation (using two Map operators in Cleenex) that splits the strings containing several author names stored in the *Authors* column of the *PublicationAuthors* table into several records, and assigns an identifier to each name (*Author ID*). A QC to detect tuples whose author name contains either “Others” or “and” was previously specified. A MDR is defined with a view over the blamed tuples generated by violations of this QC, and allows the user to delete or update the author name column value.

In Figure 5.2, we show the different iterations required to completely clean the data where a User executes this data cleaning process and manually repairs the incorrect author names that are either named “Others” or “and”. The mentioned iterative data cleaning process runs as follows:

- At iteration t, a User executes a data cleaning graph that, given the *PublicationAuthors* table as input, creates an *Authors* table where *Author ID* is the identifier of an author and *Name* is its name The tuples 1, 2, and 5 of the *Authors* table are blamed tuples. .
- At iteration $t+1$, using the MDR, the user manually repairs the tuples 1, 2, and 5 in table *Authors*. MDR *Instance ID*$= 0$ corresponds to deleting the tuple with the “and” value (tuple 1), MDRs *Instance ID*$= 1$ *and 2* correspond to replace the “Others” by values “Reiiken” and “Heiken”, respectively. The MDR instances are then applied to the Authors table resulting in a clean table with no “Others” and “and” values.
- At iteration $t+2$, the User re-executes the data cleaning graph because the original *PublicationAuthors* table was modified (e.g., new data was added, *Pub ID*$= 0$). This time, the identifier of

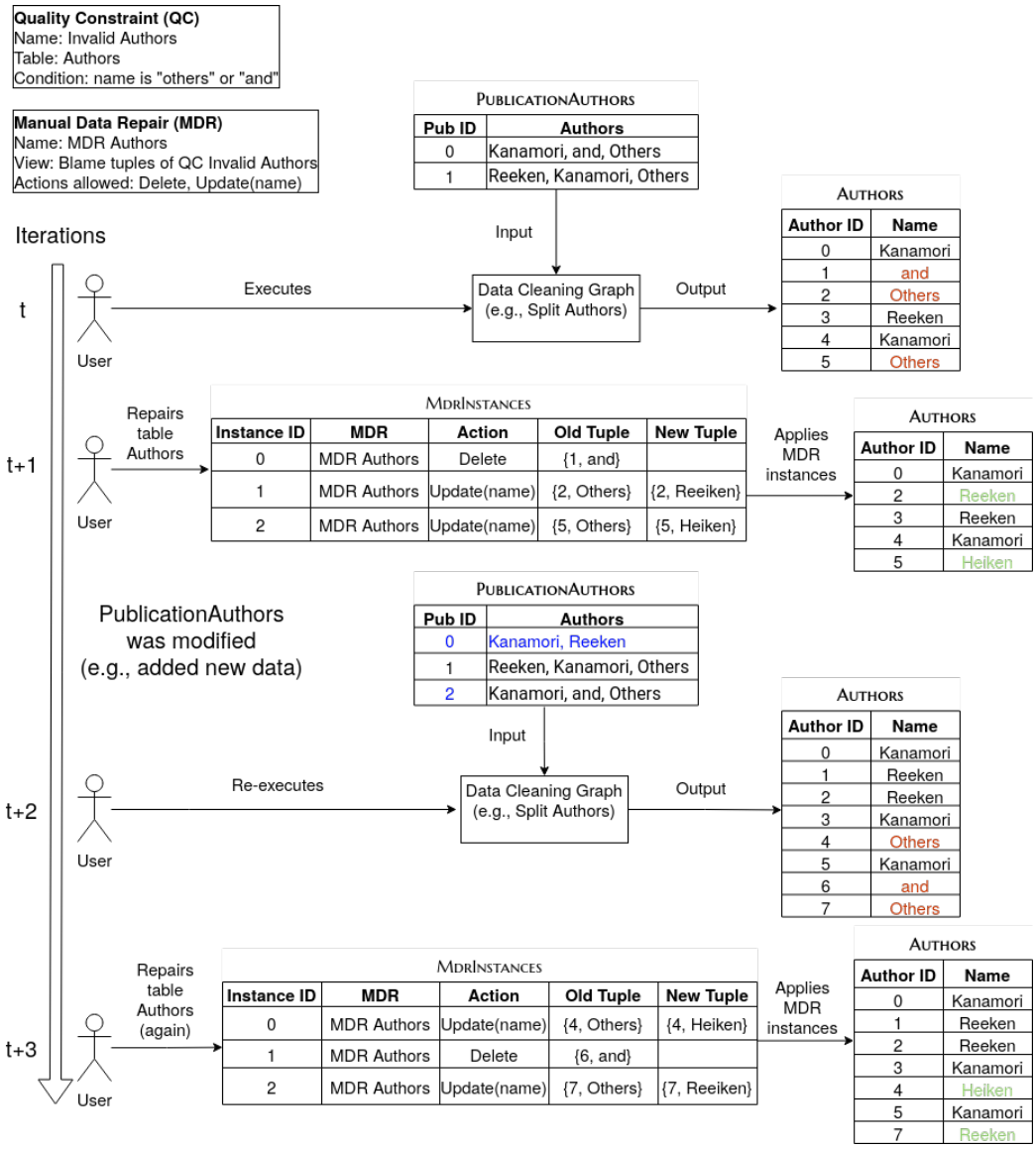


Figure 5.2: Example of user feedback in Cleenex during a data cleaning process.

the publication with original *Pub ID*$= 0$ is modified to *Pub ID*$= 2$. The result is once again a dirty *Authors* table containing “Others” and “and” values in the name column.

- At iteration $t + 3$, the User once again manually repairs the table *Authors* as she/he did in iteration $t + 1$, leading to a clean *Authors* table.

In Section 5.1.1, we detail the Cleenex software architecture; and in Section 5.1.2, we focus on the Cleenex components that support the iterative execution of a data cleaning process.

5.1.1 Cleenex architecture

The Cleenex architecture is presented in Figure 5.3, that shows the different core components of Cleenex and their interactions during an execution of a data cleaning process:

- **Processing a program specification:** Colored with green in Figure 5.3, the *Program Parser*

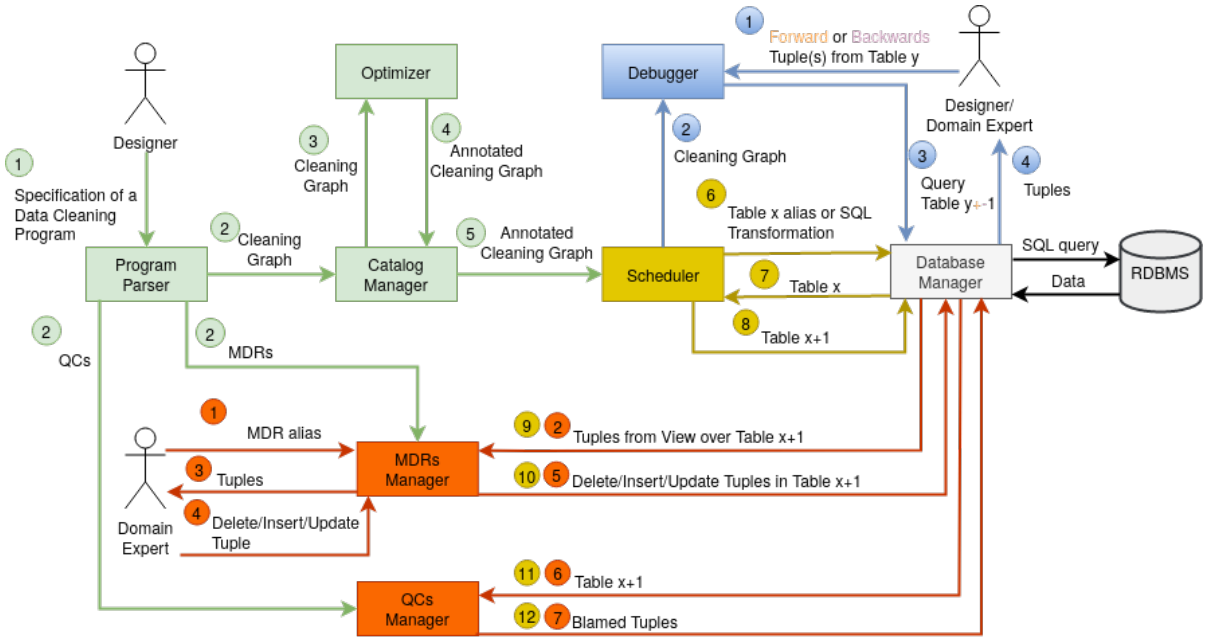


Figure 5.3: Cleenex architecture. Each color represents a different set of components and steps: green for program parsing and creation of internal representations; yellow for the data cleaning program (re-execution; red for the manual data repairs produced by a Domain Expert; blue for the use of debugger to discover the provenance of data tuples; gray for the low level components and interactions with a relational database.

converts the input specification of a data cleaning program (modeled as a data cleaning graph), QCs and MDRs using a SQL like language, into an internal representation consisting of Java objects. Then, the Program Parser sends the data cleaning graph to the Catalog Manager, the QCs to the QC Manager, and the MDRs to the MDR Manager. The *Catalog Manager* is responsible for keeping the data cleaning graph internal representation, and also interacting with the Optimizer to obtain in the annotated cleaning graph, the execution order of the data transformations in the graph and the implementations of those data transformations, either SQL or Java algorithms. The *Scheduler* receives, from the Catalog Manager, the annotated data cleaning graph.

- **Executing a data cleaning program:** Colored with yellow in Figure 5.3, the Scheduler follows the execution order specified in the annotated cleaning graph and executes the data transformations. Each data transformation is either executed as SQL or Java algorithm implementations. If it is a SQL implementation then it is executed through the Database Manager that calls the Relational DataBase Management System (RDBMS); otherwise it requests the transformation input data by providing a Table x alias through the Database Manager, executes the corresponding Java algorithm code internally, and sends the resulting data, Table $x + 1$, to the Database Manager to be stored. The *Database Manager* in gray is the Cleenex interface to communicate with the Relational Database Manager (RDBMS). Additionally, for each table in the graph, Table $x + 1$, Cleenex applies (if exists) the MDR instances and check for blamed tuples using QCs. First, the MDRs Manager receives the tuples from Table $x + 1$ to apply the MDR instances to and sends the Delete/Insert/Update Tuples to the Database Manager to update Table $x + 1$. After, the QCs

Manager analyzes the data tuples in Table $x+1$ and creates the blamed tuples which are store by the Database Manager. Only when this process finishes, the Scheduler moves to the next data transformation in the graph.

- **Applying manual data repairs:** Colored with red in Figure 5.3, a Domain Expert requests the MDR view for a Table $x+1$ using the MDR alias which is executed by the MDRs Manager. The MDRs Manager obtains the tuples corresponding to the view from the Database Manager and returns those tuples back to the Domain Expert. The Domain Expert inspects the tuples and perform a set of actions, allowed by the MDR, that can be either delete a tuple, insert a new tuple or update the values of an existing one. Those actions are received by the MDRs Manager which creates MDR instances and requests the Database Manager to update the Table $x+1$ with those actions. Finally, the QCs Manager receives the updated Table $x+1$ and applies the specified QCs which potentially produce more blamed tuples.

- **Debugging data tuples:** Colored with blue in Figure 5.3, the user requests to the Debugger the visualization of tuples that originated a specific tuple (backward) or tuples that were produced by executing a data transformation over a given tuple (forward). The *Debugger* obtains the data cleaning graph from the Scheduler and creates a query that, based on the primary keys of the tuples in a Table y, finds the tuples that originated those (backward) in table $y-1$ or were produced (forward) by those in table $y+1$. The Database Manager executes the query against the RDBMS and returns the expected tuples to the user.

5.1.2 Supporting the Iterative Execution of a Data Cleaning Process

The execution of data cleaning process is typically iterative mainly due to the following two cases: (i) the design of a data cleaning process is an iterative process where the designer refines the data cleaning graph, executes it, and then evaluates the quality of data produced; and (ii) the initial input data may change, e.g., when the amount of data is very large, it is more efficient to use a data sample (i.e., a small set of data) to design the data cleaning process. After the process is refined, the designer executes the data cleaning process to the whole data set. Due to those two cases, Cleenex may produce new data in different points of the graph, e.g., the new *PublicationAuthors* table in Figure 5.2 iteration $t+2$.

Every time a data cleaning process reruns, Cleenex automatically reapply the actions contained in MDR instances produced in a previous run to the relational data tables in the graph; otherwise, the user has to perform those actions again as in Figure 5.2 iteration $t+3$.

There are two problems with the automatic reapplication of MDR instances. First, the MDR instances have to persist in disk, so that they are not lost when a program re-executes and the MDR instances can be immediately applied if there are no changes to the data. Second, the reapplication of MDR instances to modified and/or additional data as explained above is a challenge because the data tuples may not hold all their original attribute values. For example, in Figure 5.2 iteration $t+2$, the incorrect author names, i.e., ”and“ and ”Others“ values, have now different *Author ID* values, so Cleenex is not

able to reapply the data modification to the right tuple. We define that a *MDR instance conflict* arises when Cleenex cannot automatically apply a MDR instance due to considerable data changes that do not guarantee that the modifications still hold. For example, Cleenex is not able to find which MDR instance needs to be applied to each "Others" value in Figure 5.2 iteration $t + 2$.

To support the iterative execution of a data cleaning process with the above constraints, we first added to Cleenex the *MDR Persistence* component that prevents the loss of previous MDR instances when the data cleaning process is re-executed with the same data. Then, we created the *MDR Recovery* component that tries to reapply MDR instances in case of conflict or asks the user in a concise way for additional feedback. To develop such component, in [74], the author introduced the notion of deterministic attributes in a Cleenex data cleaning graph specification. *Deterministic attributes* are attributes whose values do not change during the iterative execution of a data cleaning process (e.g., column *Name* of table *Authors* in Figure 5.2). So, only the values of deterministic attributes are taken into account to reapply MDR instances, while the remaining values are ignored because they may have changed. In the context of this component, we formulated and implemented an algorithm (detailed in Section 5.1.2.1) for the detection and automatic recovery of MDR instances[4], that tries as much as possible with 100% guarantees to recover the MDR instances from a conflict. If it is not possible to recover, the algorithm calls the user to help selecting the right data tuples to which the MDR instance should be reapplied.

Example 5.1.3. Iterative data cleaning process. To exemplify the usefulness of the MDR Persistence and the MDR Recovery components, we revisit the example of Figure 5.2 - recreating iterations $t + 2$ and $t + 3$ in Figure 5.4 using these two components:

- At iteration $t + 2$, the generated dirty *Authors* table is passed as input to the MDR Recovery component. The MDR Recovery component obtains the MDR instances created in iteration $t + 1$ through the MDR Persistence component, that retrieves them from the database. For *Instance ID*$= 0$, it can recover the tuple because there is only one "and" and it is placed in a deterministic column, so it removes the tuple. Since there are two tuples with "Others", the component is not able to identify which MDR instance to apply to each, so it creates a conflict for MDR instances 1 and 2 saving the candidate tuples Primary Keys (PK) values (in this example, the Primary Key is the *Author ID* column).

- At iteration $t + 3$, the User selects one conflict to resolve. She/he selects the MDR *Instance ID*$= 1$ and selects the tuple with *Author ID*$= 7$ to which this MDR instance should apply to. The MDR instance is updated with the new tuple and it is applied to the *Authors* table, leaving only one tuple with the value "Others". The MDR recovery component is called with the new MDR instances and the new *Author* table. Now, because only one tuple contains the "Others" value, it is able to assign it to MDR *Instance ID*$= 2$, then updates the MDR instance and applies it to the *Authors* table resulting in a cleaned table. Note that, although the number of iterations is the same, the User only had to select the tuple to which an MDR instance should be applied while before, as illustrated in Figure 5.2, the User had to produce the MDR instances from scratch.

[4]The thesis [74] proposed the first version of the algorithm and developed the support for conflict resolution in the graphical user interface of Cleenex.

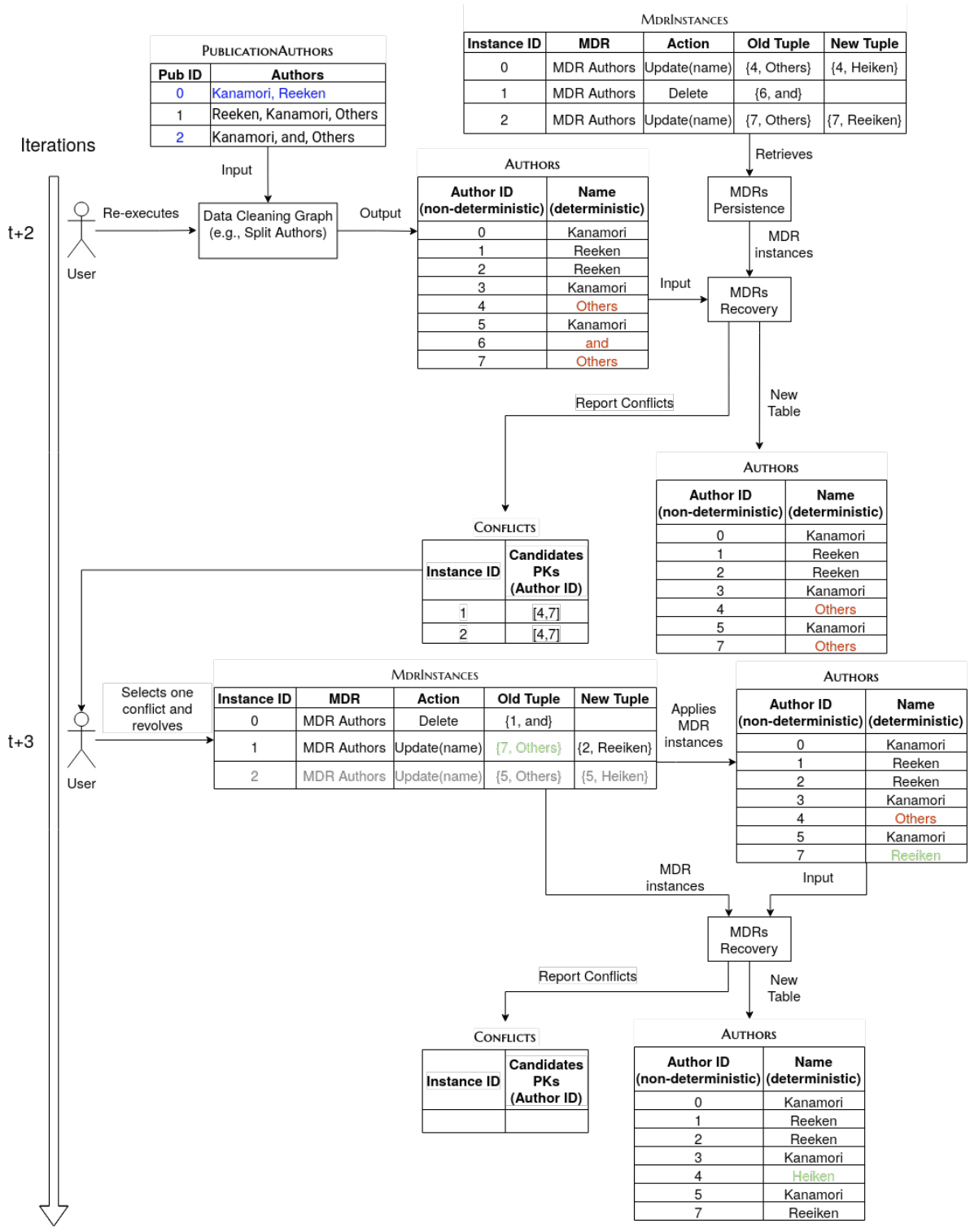


Figure 5.4: Example of user feedback in Cleenex using the MDR Persistence and the MDR recovery components during an iterative data cleaning process.

5.1.2.1 Conflict Detection and Automatic Recovery Algorithm

We present the algorithm for the conflict detection and automatic recovery of MDR instances in Algorithm 1. This algorithm receives a list of MDR instances that belong to the same MDR, applies the MDR instances that have no conflicts, and outputs a list of MDR instance conflicts to be resolved by the

user.

Algorithm 1: MDR instance recovery and conflict resolution algorithm

Input:

view = view specified in the MDR, includes the tuples in the view;

mdr_instances = list of MDR instances that belong to the same MDR;

pk_atts = list of attributes that constitute the primary key of the base relation of the view vr;

det_atts = list of attributes of the base relation of the view vr that have been declared as deterministic;

Output: conflicts = list of MDR instance conflicts

```
1  conflicts ← ∅;
2  foreach inst ∈ mdr_instances do
3      if inst.action = insert then
           /* There are no MDR Conflicts                                  */
4          view ← view ∪ inst.new_tuple;
5      else if pk_atts ⊆ det_atts then
           /* There are no MDR Conflicts                                  */
6          applyMDRInstance (inst, pk_atts, view);
7      else
8          candidate_tuples ← {tuple ∈ view such that tuple.att = inst.old_tuple.att for all
             att ∈ det_atts};
9          if |candidate_tuples| = 1 then
               /* There are no MDR Conflicts                              */
10             applyMDRInstance (inst, det_atts, view);
11         else
12             new_conflict ← conflict (inst, candidate_tuples);
13             conflicts ← conflicts ∪ new_conflict;
14         end
15     end
16 end
17 return conflicts;

   /* Applies an MDR instance to all tuples that for all attributes passed as
      argument, the values are equal to the instance old tuple values     */
18 Procedure applyMDRInstance (inst, atts, view):
19     foreach tuple ∈ view do
20         if tuple.att = inst.old_tuple.att for all att ∈ atts then
21             if inst.action = delete then
22                 view ← view \tuple;
23             else
                   /* Update MDR                                          */
24                 new_tuple ← tuple;
25                 foreach att ∈ inst.action.update do
26                     new_tuple.att ← inst.new_tuple.att ;
27                 end
28                 view ← view \tuple ∪ new_tuple;
29             end
30         end
31     end
```

At each new execution of the data cleaning graph, first the algorithm detects existing MDR instance conflicts by evaluating each MDR instance (Algorithm 1 line 2). If a MDR instance contains an insert action, then there is no conflict possible, and it is immediately applied (lines 3-4). If all attributes involved

in the MDR instance are deterministic then there is also no conflict, and the MDR instance can be applied (lines 5-6). For each remaining MDR instances, the algorithm checks the tuples whose deterministic attribute values are equal to the MDR instance Old Tuple values (line 8). If there is only one tuple (line 9), then the algorithm can apply the MDR instance (line 10), e.g., deleting "and" in Figure 5.4 iteration $t + 2$ and updating the remaining "Others" value in Figure 5.4 iteration $t + 3$; otherwise a new conflict has to be created (lines 12-13), e.g., the two "Others" values in Figure 5.4 iteration $t + 2$. A conflict contains the original MDR instance and the tuples found whose deterministic values are equal to the MDR instance Old Tuple values. When the algorithm finishes, the user can inspect the conflicts returned by the algorithm and decide how to resolve them.

5.2 User Involvement Evaluation

We conducted an extensive experimental validation to evaluate the support for the user involvement in Cleenex with two studies: (i) with a simulated user, we evaluated the Cleenex components regarding their support for the user feedback during a data cleaning process; and (ii) with real users, we conducted an experimental evaluation of user involvement during a data cleaning process with Cleenex against two tools typically used for data cleaning: OpenRefine and PDI. In [79], we conducted a preliminary experimental study to compare Cleenex against OpenRefine and PDI in the context of Approximate Duplicate Detection and Consolidation.

We report the research questions, experimental setup and obtained results to evaluate Cleenex support for the user feedback in Section 5.2.1. We detail our experiments with real users on Cleenex, PDI, and OpenRefine in Section 5.2.2.

5.2.1 User Feedback Support in Cleenex

The goal of this study was to evaluate the effectiveness of the different components incorporated in Cleenex (namely QCs/MDRs Managers, MDRs persistence, and MDRs recovery) in terms of their effect for reducing the user effort when manually cleaning data during a data cleaning process. Specifically, we investigate the following *research questions*:

RQ1.1: What is the impact of having QCs and MDRs in the context of a data cleaning process?

RQ1.2: What is the impact of having persistent MDR instances in the context of an iterative data cleaning process?

RQ1.3: What is the impact of having MDR conflict resolution in the context of an iterative data cleaning process?

For that purpose, we programmed an ideal user to execute a data cleaning program, refined by a designer, and to manually clean the required data during a data cleaning process to obtain 100% clean data. We used the following two datasets:

		Without Qcs/MDRs	With Qcs/MDRs
Visualization	# tuples visualized	1413	**365** (74%)
	# chars visualized	51259	**7308** (86%)
Insertion	# tuples inserted	214	**0** (100%)
	# chars inserted	2589	**0** (100%)
Update	# tuples updated	**0**	100 (N/A)
	# chars updated	**0**	939 (N/A)
Deletion	# tuples deleted	**1**	165 (-16400%)
Total	# tuples inserted/updated/deleted	**215**	265 (-23%)
	# chars inserted/updated	2589	**939** (64%)

Table 5.1: Simulated user results for executing the first iteration of the data cleaning process for the BPA dataset. Gain percentage in relation to the left column components is reported in parentheses.

		Without MDRs Persistence and MDRs Recovery	With MDRs Persistence and without MDRs Recovery	With MDRs Persistence and MDRs Recovery
Visualization	# tuples visualized	365	265 (27%)	**106** (60%)
	# chars visualized	7308	6426 (12%)	**2481** (61%)
Insertion	# tuples inserted	**0**	**0** (N/A)	**0** (N/A)
	# chars inserted	**0**	**0** (N/A)	**0** (N/A)
Update	# tuples updated	100	**0** (100%)	**0** (N/A)
	# chars updated	939	**0** (100%)	**0** (N/A)
Deletion	# tuples deleted	165	165 (0%)	**26** (84%)
Total	# tuples inserted/updated/deleted	265	165 (38%)	**26** (84%)
	# chars inserted/updated	939	**0** (100%)	**0** (N/A)

Table 5.2: Simulated user results per iteration of the iterative data cleaning process excluding the first one for the BPA dataset. Gain percentage in relation to the left column components is reported in parentheses.

1. **Big Publication Authors** (BPA) which consists in a table that lists publications and corresponding authors where some author names are written in distinct ways (approximate duplicates);

2. **Customers** (C) which contains three tables: one for customers with inconsistent phone number formats, incorrect/missing values in customer name, phone number, street and city, another table for treatments where an insurance names are written differently (approximate duplicates) and a master data table that contains a cleaned subset of customer records.

For each dataset, we evaluated the user effort needed to clean the remaining data quality problems given a data cleaning program design by an expert. For that purpose, we measured the number of tuples and characters (chars) visualized, inserted, updated and deleted that the user had to perform to obtain 100% clean data. Table 5.1 reports the results of executing the data cleaning program for the Big Publication Authors (BPA) dataset for a first iteration of the iterative data cleaning process using Cleenex with and without QCs and MDRs. In Table 5.2, we report further iterations over the same data cleaning process for BPA with QCs and MDRs. We measured the user effort when cleaning again the same data quality problems with and without the MDRs Persistence component and with and without the MDRs Recovery component. Table 5.3 reports the results of executing the data cleaning program for the Customers (C) dataset with and without QCs and MDRs.

RQ1.1. Based on the results in Tables 5.1 and 5.3, we observe that performing a data cleaning process with QCs/MDRs reduces the user effort because the user needs to visualize less data (74%-99% less tuples and 86%-99% less chars) and perform effortless manual data repairs comparatively to the same

		Without Qcs/MDRs	With Qcs/MDRs
Visualization	# tuples visualized	176200	**1739** (99%)
	# chars visualized	9902194	**145017** (99%)
Insertion	# tuples inserted	**0**	**0** (N/A)
	# chars inserted	**0**	**0** (N/A)
Update	# tuples updated	268	**69** (74%)
	# chars updated	5851	**333** (94%)
Deletion	# tuples deleted	**0**	200 (N/A)
Total	# tuples inserted/updated/deleted	**268**	269 (0%)
	# chars inserted/updated	5851	**333** (94%)

Table 5.3: Simulated user results for the C dataset. Gain percentage in relation to the left column components is reported in parentheses.

process without QCs/MDRs.

RQ1.2. The MDRs Persistence component reduced the user effort when manually data cleaning in an iterative data cleaning process. As observable in Table 5.2, the tuples updates were reduced to zero and the tuples visualized dropped by 27%.

RQ1.3. Regarding an iterative data cleaning process with the MDRs Recovery component, we observe also in Table 5.2 that this component reduces the user effort. The MDRs Recovery component was able to recover most of the manual data repair instances created by the user in previous iterations of the data cleaning process. For the cases where recovery was not possible, the MDRs Recovery component guided the user into recovering conflicting MDR instances from previous iterations with less effort (84& less deleted tuples) than creating those MDR instances again.

5.2.2 User Involvement Support in a Data Cleaning Process

In this study, we aimed at evaluating whether Cleenex effectively helps program designers and domain expert users to perform data cleaning tasks in a realistic scenario. We investigated different aspects of usability of Cleenex, OpenRefine, and PDI. Specifically, we try to answer the following *research questions*:

RQ2.1: Are Cleenex data cleaning programs easier to understand than programs specified using the other two tools used for data cleaning?

RQ2.2: Given a specification of a data cleaning program written and refined by an expert, using Cleenex requires less time/user effort to obtain cleaned data than using the other two tools used for data cleaning?

RQ2.3: Given a specification of a data cleaning program written and refined by an expert, does the data obtained with Cleenex have higher quality than the data obtained with the other two tools used for data cleaning?

RQ2.4: Does the debugging functionality help to obtain more correct data than the other two tools used for data cleaning?

The preparation of this study included: (i) a video tutorial for each tool, (ii) an experimental plan, (iii) an experimental guide, and (iv) a satisfaction questionnaire. We asked the users to perform two tasks

Group	ID	Question	Type/Scale	Research Question
User Perception (after completing Task 1)	UP1	How easy was it to understand the data cleaning program?	1- very difficult, 7- very easy	RQ2.1
	UP2	How difficult was it to answer the questions about the data cleaning program?	1- very difficult, 7- very easy	RQ2.1
	UP3	How much effort was required to answer the questions about the data cleaning program?	1- almost no effort, 7 – extreme effort	RQ2.1
Program Specification	PS1	It was easy to understand this tool operators	1 - strongly disagree, 7 – strongly agree	RQ2.1
	PS2	It was easy to understand the source of specific records using this tool	1 - strongly disagree, 7 – strongly agree	RQ2.3
Manually correcting data	MCD1	It was easy to find the records that I wanted to modify using [this tool].	1 - strongly disagree, 7 – strongly agree	RQ2.2
	MCD2	It was easy to manually repair records using [this tool].	1 - strongly disagree, 7 – strongly agree	RQ2.2
User Preference	UPF1	What would you use to specify a data cleaning program?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF2	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF3	Consider that you have the same data cleaning program implemented in both tools, which tool would you use to understand the source of specific records in a data cleaning program?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF4	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF5	Consider that you have the same data cleaning program implemented in both tools, what would you use to manually modify data?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF6	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF7	Based on your experience with Cleenex, please describe which additional feature(s) would you like to see implemented and why (you can compare with other tools)?	Open answer	Overall

Table 5.4: Perception questions asked to the user.

Group	Question	Scale
Usefulness	Using this tool to perform data cleaning tasks would enable me to accomplish tasks more quickly.	1 - strongly disagree, 7 – strongly agree
	Using this tool to perform data cleaning tasks would improve my performance (quality of output).	1 - strongly disagree, 7 – strongly agree
	Using this tool to perform data cleaning tasks would increase my productivity (efficiency of production, Output/Input).	1 - strongly disagree, 7 – strongly agree
	Using this tool would enhance my effectiveness (accomplish to do the right tasks) to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
	Using this tool would make it easier to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
	I would find this tool useful to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
Ease of use	Learning to operate with this tool would be easy for me.	1 - strongly disagree, 7 – strongly agree
	I would find it easy to get this tool to do what I want it to do.	1 - strongly disagree, 7 – strongly agree
	My interaction with this tool would be clear and understandable.	1 - strongly disagree, 7 – strongly agree
	I would find this tool to be flexible to interact with.	1 - strongly disagree, 7 – strongly agree
	It would be easy for me to become skillful at using this tool.	1 - strongly disagree, 7 – strongly agree
	I would find this tool easy to use.	1 - strongly disagree, 7 – strongly agree

Table 5.5: TAM satisfaction questionnaire questions asked for the user.

and answer a set of questions before, between, and after performing these tasks in order to access their perception. The satisfaction questionnaire included: (i) questions about the user perception, as listed in Table 5.4 distinguished by the Research Question identifier they are intend to help answer or classified as Overall if a general question about user preference, and (ii) the questions from the standard Technology Acceptance Model (TAM) questionnaire [27] as in Table 5.5. TAM questions are grouped into usefulness and ease of use. The evaluation of these TAM questions is usually performed by analyzing the sum of the scores obtained for each group.

The first task, *Task 1*, contains several questions that evaluate the user's understandability regarding one of the data cleaning programs designed by an expert. We evaluated the score obtained for each understandability question to measure how easier is to understand a program. The second task, *Task 2*, uses the same data cleaning program and requires the user to manually clean data in order to obtain 100% clean data. We measured the final output data quality in terms of Precision, Recall and F1-Measure. During the execution of Task 1 and Task 2, we measured the number of clicks and keys pressed by the user, as well as the time to complete each task and the number of times the Cleenex

			Cleenex Vs OpenRefine			Cleenex Vs PDI			Experts – PDI Vs Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Task1 - Understandability	Cleenex	Task 1 # Correct Answers	7.67	8.00	1.25E-04	6.92	7.00	1.21E-06	6.88	7.00	1.05E-06
	Other Tool	Task 1 # Correct Answers	5.92	6.00	5.75E-01	5.00	5.00	5.98E-01	6.50	7.00	6.32E-05
	Tools Difference	Task 1 # Correct Answers	1.75	2.00	3.91E-03	1.92	2.00	9.77E-04	0.38	0.00	1.97E-01
User Effort	Cleenex	Task1 # Clicks	57.92	38.00	5.76E-03	39.58	36.50	7.11E-01	58.88	53.00	1.83E-01
	Other Tool	Task1 # Clicks	117.08	78.00	3.78E-04	171.00	154.00	7.15E-01	116.38	85.00	4.61E-03
	Tools Difference	Task1 # Clicks	-59.17	-29.00	3.27E-02	-131.42	-120.50	4.88E-04	-34.38	-37.00	1.48E-01
	Cleenex	Task1 # Keys	0.00	0.00	1.00E-05	0.00	0.00	1.00E-05	0.38	0.00	4.79E-04
	Other Tool	Task1 # Keys	5.92	0.00	2.78E-05	3.83	1.00	1.06E-04	2.38	0.00	1.52E-05
	Tools Difference	Task1 # Keys	-5.92	0.00	1.54E-01	-3.83	-1.00	9.76E-02	0.25	0.00	3.57E-01
User Perception (after completing the task)	Cleenex	Answers to UP1	5.67	6.00	2.28E-02	5.83	6.00	2.90E-03	5.38	5.00	3.24E-01
	Other Tool	Answers to UP1	4.08	4.00	2.11E-01	5.58	6.00	5.88E-02	5.00	5.00	1.20E-01
	Tools Difference	Answers to UP1	1.58	1.50	3.91E-03	0.25	0.00	5.63E-01	0.38	0.50	5.31E-01
	Cleenex	Answers to UP2	5.58	5.50	1.18E-01	5.42	6.00	2.81E-02	5.88	6.00	3.70E-02
	Other Tool	Answers to UP2	3.42	3.00	1.52E-02	4.00	4.00	2.59E-01	5.13	5.00	1.95E-01
	Tools Difference	Answers to UP2	2.17	2.00	4.88E-04	1.42	1.50	6.54E-02	0.75	1.00	2.19E-01
	Cleenex	Answers to UP3	3.00	2.50	6.57E-02	2.92	2.50	1.95E-02	2.63	2.50	2.45E-01
	Other Tool	Answers to UP3	4.50	5.00	1.73E-02	4.33	5.00	5.89E-01	3.25	3.00	4.08E-01
	Tools Difference	Answers to UP3	-1.50	-1.50	3.81E-02	-1.42	-1.50	3.13E-02	-0.63	-0.50	3.13E-01
Easiness User Perception	Cleenex	Answers to PS1	5.25	6.00	2.72E-02	6.08	6.00	1.52E-02	5.25	5.50	1.85E-02
	Other Tool	Answers to PS1	3.42	3.50	1.89E-01	4.92	5.50	9.98E-02	5.13	5.50	3.56E-02
	Tools Difference	Answers to PS1	1.83	1.50	4.10E-02	1.17	0.50	9.38E-02	0.13	0.00	7.85E-01
	Cleenex	Answers to PS2	5.67	6.00	2.28E-02	6.33	6.50	5.21E-03	6.63	7.00	4.79E-04
	Other Tool	Answers to PS2	3.25	3.00	4.95E-01	4.58	5.50	5.88E-02	5.25	6.00	7.45E-02
	Tools Difference	Answers to PS2	2.42	2.00	9.77E-04	1.75	1.50	4.69E-02	1.38	1.00	6.25E-02

Table 5.6: Research metrics and answers related to RQ2.1.

debugger was used. Users had to perform both tasks using Cleenex and using another tool. So, we used two datasets with the corresponding data cleaning program: (i) **Childhood Locations** (CL) based on a real dataset that lists Chicago early childhood locations that contains approximate duplicated records; and (ii) **Publication Authors** (PA) consists in a table that lists publications and corresponding authors where some author names are written in distinct ways (approximate duplicates).

There were 3 groups of users: (i) users who performed experiments on Cleenex and OpenRefine, (ii) users who performed experiments on Cleenex and PDI, and (iii) users who typically use PDI in their work, and performed the experiments using Cleenex and PDI. The regular participants (i.e., user groups (i) and (ii)) in this study were 24 (12 in each group) and the PDI experts (i.e., user group (iii)) were eight in total.

We organized the different results per research question. We report the results for RQ2.1, RQ2.2, RQ2.3, and RQ2.4 respectively in Tables 5.6, 5.7, 5.8, and 5.9. In addition, we report overall results regarding the TAM scores and user preferences in Table 5.10. For each tool and for each question score and measure, we report the average, the median of the values and the P-value. The P-value corresponds to a normality test, namely Shapiro Test. We consider that the variable (i.e., the results for a given tool) follow a normal distribution if P-value $<= 0.05$. For each question score and measure, when possible we also report the difference between the values obtained for Cleenex and for another tool in a column identified as Tools Difference. For the difference, we report the average, the median and the P-value that corresponds to a pair test over the values obtained on both tools. If both testing variables (measure values obtained with Cleenex and another tool) are normally distributed, we applied the Paired t-test, otherwise we applied the Wilcoxon Test. We statistically accept that Cleenex is better in a particular measure or question if the test passes with P-value $<= 0.05$.

RQ2.1. For the RQ2.1, Cleenex did not pass the statistical test against OpenRefine in Task 1 execution time and number of keys pressed, still, the participants took less time and used less clicks. Both user perception and score passed the test. Regarding Cleenex against PDI, we observe that the score, UP3,

			Cleenex Vs OpenRefine			Cleenex Vs PDI			Experts – PDI Vs Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
User Effort	Cleenex	Task2 Execution Time (s)	636.50	576.50	9.44E-01	248.83	199.50	1.70E-03	234.00	208.50	2.61E-02
	Other Tool	Task2 Execution Time (s)	850.42	834.00	9.28E-01	508.75	423.50	9.42E-03	586.88	526.00	4.38E-02
	Tools Difference	Task2 Execution Time (s)	-180.50	-142.00	4.88E-04	-259.92	-296.00	1.72E-02	-352.88	-336.00	1.99E-02
	Cleenex	Task2 # Clicks	52.08	44.00	3.37E-02	14.67	11.50	4.99E-02	21.38	21.50	3.56E-01
	Other Tool	Task2 # Clicks	113.50	101.00	1.06E-01	72.42	71.50	2.68E-01	116.38	85.00	4.61E-03
	Tools Difference	Task2 # Clicks	-56.58	-47.00	1.46E-03	-57.75	-58.50	1.95E-03	-95.00	-63.50	7.81E-03
	Cleenex	Task2 # Keys	0.17	0.00	1.21E-06	0.00	0.00	1.00E-05	0.00	0.00	1.00E-05
	Other Tool	Task2 # Keys	14.67	6.00	9.62E-04	28.17	1.00	1.52E-04	2.38	0.00	1.52E-05
	Tools Difference	Task2 # Keys	-14.17	-6.00	3.25E-02	-28.17	-1.00	9.67E-02	-2.38	0.00	2.70E-01
User Effort Perception	Cleenex	Answers to MCD1	5.58	6.00	2.80E-02	6.33	7.00	1.05E-03	6.25	6.50	1.85E-02
	Other Tool	Answers to MCD1	3.17	3.50	6.05E-02	3.92	4.00	5.50E-01	3.63	3.50	1.62E-01
	Tools Difference	Answers to MCD1	2.42	2.50	9.77E-03	2.42	2.50	5.86E-03	2.63	3.00	1.56E-02
	Cleenex	Answers to MCD2	5.25	6.00	8.92E-03	6.50	7.00	1.69E-04	7.00	7.00	1.00E-05
	Other Tool	Answers to MCD2	3.17	3.00	8.69E-02	3.75	3.50	8.93E-01	5.25	5.50	3.34E-01
	Tools Difference	Answers to MCD2	2.08	2.00	2.88E-02	2.75	3.00	3.91E-03	1.75	1.50	3.13E-02

Table 5.7: Research metrics and answers related to RQ2.2.

			Cleenex Vs OpenRefine			Cleenex Vs PDI			Experts – PDI Vs Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Task 2 – Output Quality	Cleenex	Precision	0.83	1.00	9.81E-06	1.00	1.00	1.00E-05	1.00	1.00	1.00E-05
	Other Tool	Precision	0.51	0.50	3.52E-02	0.94	1.00	4.40E-05	1.00	1.00	1.00E-05
	Tools Difference	Precision	0.32	0.25	1.06E-02	0.06	0.00	8.19E-02	0.00	0.00	1.00E+00
	Cleenex	Recall	0.79	1.00	6.71E-05	1.00	1.00	1.00E-05	1.00	1.00	1.00E-05
	Other Tool	Recall	0.71	1.00	2.34E-04	0.94	1.00	1.21E-06	0.96	1.00	1.05E-06
	Tools Difference	Recall	0.08	0.00	1.66E-01	0.06	0.00	3.39E-01	0.04	0.00	3.51E-01
	Cleenex	F1-Measure	0.81	1.00	4.55E-05	1.00	1.00	1.00E-05	1.00	1.00	1.00E-05
	Other Tool	F1-Measure	0.58	0.67	2.80E-02	0.92	1.00	9.77E-05	0.98	1.00	1.05E-06
	Tools Difference	F1-Measure	0.23	0.17	1.18E-02	0.08	0.00	9.65E-02	0.03	0.00	3.51E-01

Table 5.8: Research metrics and answers related to RQ2.3.

			Cleenex Vs OpenRefine			Cleenex Vs PDI			Experts – PDI Vs Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Debugger Usage	Cleenex	# Debugger uses in Task1	1.58	1.50	6.93E-02	1.67	1.50	5.21E-03	2.75	2.00	1.77E-05
	Cleenex	# Debugger uses in Task2	0.00	0.00	1.00E-05	0.00	0.00	1.00E-05	0.00	0.00	1.00E-05
User Perception	Cleenex	Answers to PS2	5.67	6.00	2.28E-02	6.33	6.50	5.21E-03	6.63	7.00	4.79E-04
	Other Tool	Answers to PS2	3.25	3.00	4.95E-01	4.58	5.50	5.88E-02	5.25	6.00	7.45E-02
	Tools Difference	Answers to PS2	2.42	2.00	9.77E-04	1.75	1.50	4.69E-02	1.38	1.00	6.25E-02
	Overall	Answers to UPF3	6.08	6.00	2.75E-02	6.58	7.00	4.82E-04	1.38	1.00	4.45E-04

Table 5.9: Research metrics and answers related to research RQ2.4.

			Cleenex Vs OpenRefine			Cleenex Vs PDI			Experts – PDI Vs Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
TAM	Cleenex	Sum TAM Usefulness Answers	34.67	36.00	6.02E-01	39.67	40.00	1.37E-02	33.00	32.00	N/A
	Other Tool	Sum TAM Usefulness Answers	23.00	23.00	5.19E-01	29.67	32.00	3.00E-01	N/A	N/A	N/A
	Tools Difference	Sum TAM Usefulness Answers	11.67	11.00	3.91E-03	10.00	8.00	5.86E-03	N/A	N/A	N/A
	Cleenex	Sum TAM Ease of Use Answers	35.08	36.00	2.06E-01	37.08	38.00	6.14E-01	31.50	33.50	N/A
	Other Tool	Sum TAM Ease of Use Answers	23.33	23.00	8.84E-02	28.75	26.00	3.35E-01	N/A	N/A	N/A
	Tools Difference	Sum TAM Ease of Use	11.75	10.50	9.77E-04	8.33	6.00	3.91E-03	N/A	N/A	N/A
User Preference	Overall	Answers to UPF1	6.17	6.00	1.53E-02	5.75	6.00	3.80E-02	4.13	4.00	4.37E-02
		Answers to UPF2	6.08	6.00	2.75E-02	6.58	7.00	4.82E-04	6.00	6.50	4.45E-04
		Answers to UPF5	5.75	6.00	1.72E-02	6.42	7.00	1.64E-03	5.25	6.50	1.07E-02

Table 5.10: Research metrics and answers related to overall analysis and user preferences.

PS2 and the number of clicks pass the test, the other metrics indicate that Cleenex was better than PDI, but the difference was not statistically significant. In the experiments with PDI experts, we could not statistically prove any metric because the P-values of the Tools Differences are higher than 0.05. Nevertheless, experts performed similar or better on Cleenex, and answer the questions positively to Clennex against PDI.

RQ2.2. Regarding RQ2.2, almost all metrics in all types of experiments (Cleenex vs OpenRefine, Cleenex vs PDI, and PDI experts) are statistically proved. The exception is the number of keys pressed

for Cleenex vs PDI and PDI experts that did not pass the test. Still, participants on Cleenex pressed less keys than in other tool.

RQ2.3. RQ2.3 depends on the output data quality resulting from Task 2. Only when we compare Cleenex against OpenRefine we can statistically prove the results of Precision and F1-Measure. However, we can observe that participants using Cleenex got better quality than using another tool even among the PDI experts.

RQ2.4. For RQ2.4, we observe that the debugger was used on Task 1 but was never used on Task 2. Regarding PS2, we verify with statistically significant that users consider easier to find the source of a record using Cleenex instead of OpenRefine and PDI. Among PDI experts, we also got a better PS2 but without statistically proof. Overall, UPF3, participants (including PDI experts) preferred Cleenex to other tool.

Based on the obtained results:

- We concluded that:
  - Cleenex data cleaning programs are easier to understand than programs specified using the other considered tools;
  - Given a specification of a data cleaning program written and refined by an expert, Cleenex requires less time/user effort to obtain cleaned data than another tool;
  - The debugging functionality helps to obtain more correct data than using the other tools.
- Still, with no statistical proof but with positive results, we observed that:
  - Given a specification of a data cleaning program written and refined by an expert, the data obtained with Cleenex has higher quality than using any of the other two tools.
- The user answer values to the TAM [27] questionnaire and to the quantitative Overall questions on User Preference (UPF1, UPF3, and UPF5) about user tool preference are greater for Cleenex than for OpenRefine and PDI, and when applicable are statistically significant. Those answers indicate that users prefer to perform data cleaning tasks with Cleenex than with any of the other two tools.
- For the open answers UPF2, UPF4, UPF6, and UPF7, the users highlight Cleenex components that help in designing the data cleaning program and manual cleaning data, such as the graph and code editors, the debugger, and the Manual Data Repairs (MDRs) that do not exist in OpenRefine and PDI.

In summary, from both experimental studies, we conclude that the new developed Cleenex successfully helps reducing the user effort to effectively clean data during a data cleaning process.

Chapter 6

Conclusions

In this Chapter, we summarize this thesis work and the resulting main contribution in Section 6.1, and we describe and discuss possible directions for future research works in Section 6.2.

6.1 Summary and Main Contributions

In this thesis, we contributed to two important subjects for data cleaning: (i) acronym expansion to expand acronyms found in text documents and (ii) support the user involvement during the execution of a data cleaning program.

Specifically, we described our main contributions in acronym expansion: (i) the end-to-end Acronym eXpander (AcX) system that expands acronyms found in text; (ii) three benchmarks to compare in- and out- expansion techniques and end-to-end systems; (iii) a new dataset constituted by in and out expansions used to evaluate the end-to-end systems. The AcX system synthesizes and extends the best of previous work on acronym expansion.

In the process, our major technical findings are:

- In-expansion rule-based techniques (SH and MadDog-in) usually work best and require little execution time.
- For out-expansion, SciDr-out and Cosine similarity (Cossim) or Support Vector Machines (SVMs) with SBERT usually work best , followed by Cosine similarity (Cossim) and Support Vector Machines (SVMs) with either Classic Context Vector (CCV) or Doc2Vec.
- There is still a significant gap between the best AcX pipelines and human-level performance.

There are five data and software products of our work that future researchers can either extend or use as a basis of comparison.

1. The first human-annotated dataset for end-to-end acronym expander systems.
2. Three benchmarks to evaluate: (i) in-expansion techniques, (ii) out-expansion techniques, (iii) the combination in an end-to-end setting.

3. The end-to-end AcX system is available publicly and can be applied to arbitrary languages, follows hyperlinks, and can incorporate new in- and out-expansion techniques.

In order to support the user involvement during the execution of a data cleaning program, we performed the following main contributions:

1. Cleenex, a data cleaning framework, developed into a stable version.
2. A conflict detection and automatic recovery of MDR instances algorithm, and components that enable the iterative execution of data cleaning programs with automatic re-application of manual data repair.
3. An extensive user involvement evaluation that includes two studies:
   (a) The evaluation of the Cleenex components that support user feedback in a data cleaning process with a simulated user.
   (b) The evaluation of the user involvement support in a data cleaning process of Cleenex, OpenRefine, and Pentaho Data Integration (PDI) with real users.

When evaluating individually the Cleenex components that support the incorporation of user feedback using a simulated user, we verified that performing a data cleaning process with the full set of Cleenex components (i.e., Quality Constraints, Manual Data Repairs (MDRs), MDRs persistence, and MDRs recovery), it reduces the user effort when cleaning data.

When evaluating Cleenex against the other two tools (OpenRefine and PDI) regarding user involvement, we made the following main conclusions:

1. Understand data cleaning programs is easier if specified in Cleenex.
2. The use of Cleenex lead to less time and effort to obtain clean data.
3. Users have a higher preference to use Cleenex to perform a data cleaning task.
4. Users highlight the main innovations of Cleenex such as the Manual Data Repairs (MDRs), and the Debugger that the other tools used in the study do not support.

6.2 Future Work

For acronym expansion, because the automated techniques in the state-of-the-art fall well below human-level accuracy levels, there is a large margin for improvement. We see the need for improvements in both in-expansion (especially acronym identification) and out-expansion. Some promising avenues for improvements include: (i) more accurate in-expansion (e.g., additional acronym-expansion extraction patterns), (ii) new context representation techniques, and (iii) an extensive study of ensemble techniques.

Since AcX easily extends to other languages (e.g., our Portuguese extension was done by a high school student), we think it would be interesting to create AcX pipelines for a variety of natural languages.

To further improve the support of user feedback in Cleenex, the next big research task would be to introduce Machine Learning techniques that identify and automate cleaning patters that the user can provide during the execution of the data cleaning process. Such techniques, would replace the user feedback given on large quantities of data and conduct a still effective data cleaning process. Furthermore, we consider important to research the efficient support of concurrent user feedback, when multiple experts clean data during the same data cleaning process.

In this thesis, we strived to provide new tools for researchers and general public to improve their readability of text documents and efficiently improve data quality. The code and data produced during this work can be extended and further developed for new research works in data cleaning.

Bibliography

[1] ABBREX. ABBREX - The Abbreviation Expander, 2011. URL http://abbrex.com/.

[2] Khaled Abdalgader and Andrew Skabar. Unsupervised Similarity-based Word Sense Disambiguation Using Context Vectors and Sentential Word Importance. *ACM Trans. Speech Lang. Process.*, 9(1):2–21, 2012. ISSN 1550-4875. doi: 10.1145/2168748.2168750. URL http://doi.acm.org/10.1145/2168748.2168750.

[3] Hiroko Ao and Toshihisa Takagi. Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586, 2005.

[4] Ahmad Assadi, Tova Milo, and Slava Novgorodov. Cleaning Data with Constraints and Experts. Technical report, 2017. URL http://slavanov.com/research/dance-tr.pdf.

[5] Ahmad Assadi, Tova Milo, and Slava Novgorodov. Cleaning Data with Constraints and Experts. In *International Workshop on the Web and Databases*, WebDB'18, pages 1:1—-1:6, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5648-0. doi: 10.1145/3201463.3201464. URL http://doi.acm.org/10.1145/3201463.3201464.

[6] Giusepppe Attardi. Wikiextractor. https://github.com/attardi/wikiextractor, 2015.

[7] José Barateiro and Helena Galhardas. A survey of data quality tools. *Datenbank-Spektrum*, 14 (15-21):48, 2005.

[8] Catriel Beeri and Moshe Y Vardi. A Proof Procedure for Data Dependencies. *Journal of the ACM*, 31(4):718–741, 1984. ISSN 0004-5411. doi: 10.1145/1634.1636. URL http://doi.acm.org/10.1145/1634.1636.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: Pretrained Language Model for Scientific Text. In *Empirical Methods in Natural Language Processing*, 2019.

[10] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. Query-Oriented Data Cleaning with Oracles. In *Special Interest Group on Management of Data*, pages 1199–1214, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-2758-9. doi: 10.1145/2723372.2737786. URL http://doi.acm.org/10.1145/2723372.2737786.

[11] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[13] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *International Conference on Data Engineering*, pages 746–755. IEEE, 2007. ISBN 1424408032.

[14] Loreto Bravo, Wenfei Fan, and Shuai Ma. Extending dependencies with conditions. *Proceedings of the Very Large Data Bases Endowment*, pages 243–254, 2007.

[15] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[16] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, apr 1998. ISSN 0169-7552. doi: 10.1016/S0169-7552(98)00110-X. URL https://www.sciencedirect.com/science/article/pii/S016975529800110X.

[17] Jean Charbonnier and Christian Wartena. Using Word Embeddings for Unsupervised Acronym Disambiguation. In *International Conference on Computational Linguistics*, pages 2610–2619, Santa Fe, New Mexico, USA, 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1221.

[18] Fei Chiang and Renee J Miller. A Unified Model for Data and Constraint Repair. In *International Conference on Data Engineering*, pages 446–457, Washington, DC, USA, 2011. IEEE. ISBN 978-1-4244-8959-6. doi: 10.1109/ICDE.2011.5767833. URL http://dx.doi.org/10.1109/ICDE.2011.5767833.

[19] Peter Christen. Febrl – A Freely Available Record Linkage System with a Graphical User Interface. In *Second Australasian Workshop on Health Data and Knowledge Management*, 2008. ISBN 9781920682613. doi: 10.1051/lait:1997648.

[20] Manuel R Ciosici and I Assent. Abbreviation Expander - a Web-based System for Easy Reading of Technical Documents. In *COLING*, 2018.

[21] Manuel R. Ciosici, Tobias Sommer, and Ira Assent. Unsupervised abbreviation disambiguation contextual disambiguation using word embeddings. *Computing Research Repository*, arXiv:1904.00929, 2019. URL http://arxiv.org/abs/1904.00929. version 2.

[22] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[23] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. Improving Data Quality: Consistency and Accuracy. *Proceedings of the Very Large Data Bases Endowment*, pages 315–326, 2007. URL http://dl.acm.org/citation.cfm?id=1325851.1325890.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, arXiv:1507.07998, 2015. URL https://arxiv.org/abs/1507.07998. version 1.

[26] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. NADEEF: a commodity data cleaning system. In *Special Interest Group on Management of Data*, page 541, New York, New York, USA, 2013. ACM. ISBN 9781450320375. doi: 10.1145/2463676.2465327. URL http://dl.acm.org/citation.cfm?id=2463676.2465327.

[27] Fred D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3):319, 1989. ISSN 02767783. doi: 10.2307/249008. URL https://www.jstor.org/stable/249008?origin=crossref.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Computing Research Repository*, arXiv:1810.04805, 2018. URL https://arxiv.org/abs/1810.04805. version 1.

[29] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[30] Nicholas Egan and John Bohannon. Primer AI's Systems for Acronym Identification and Disambiguation. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org, 2021. URL http://ceur-ws.org/Vol-2831/paper30.pdf.

[31] Ronald Fagin. Functional dependencies in a relational database and propositional logic. *IBM Journal of Research and Development*, 21(6):534–544, 1977.

[32] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, may 2005. ISSN 0304-3975. doi: 10.1016/J.TCS.2004.10.033. URL https://www.sciencedirect.com/science/article/pii/S030439750400725X.

[33] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, jun 2008. ISSN 1532-4435.

[34] Wenfei Fan and Floris Geerts. *Foundations of Data Quality Management*, volume 4. Morgan & Claypool, 2012. ISBN 160845777X, 9781608457779. doi: 10.2200/S00439ED1V01Y201207DTM030. URL http://www.morganclaypool.com/doi/abs/10.2200/S00439ED1V01Y201207DTM030.

[35] Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. Dynamic constraints for record matching. *Very Large Data Bases Journal*, 20(4):495–520, 2011. ISSN 10668888. doi: 10.1007/s00778-010-0206-6.

[36] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *Very Large Data Bases*, pages 371–380. Morgan Kaufmann Publishers Inc., 2001.

[37] Helena Galhardas, Antónia Lopes, and Emanuel Santos. Support for user involvement in data cleaning. In *Data Warehousing and Knowledge Discovery*, volume 6862, pages 136–151, Toulouse, France, 2011. Springer-Verlag. ISBN 9783642235436. doi: 10.1007/978-3-642-23544-3_11.

[38] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[39] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The LLUNATIC data-cleaning framework. *Proceedings of the Very Large Data Bases Endowment*, 6(9):625–636, 2013. ISSN 21508097. doi: 10.14778/2536360.2536363. URL http://dl.acm.org/citation.cfm?doid=2536360.2536363.

[40] Mark Girolami and Ata Kabán. On an Equivalence Between PLSI and LDA. In *Special Interest Group on Information Retrieval*, pages 433–434, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: 10.1145/860435.860537. URL http://doi.acm.org/10.1145/860435.860537.

[41] Michael R. Glass, Md. Faisal Mahbub Chowdhury, and Alfio Massimiliano Gliozzo. Language independent acquisition of abbreviations. *Computing Research Repository*, arXiv:1709.08074, 2017. URL http://arxiv.org/abs/1709.08074. version 1.

[42] Phil Gooch. BADREX: in situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. *Computing Research Repository*, arXiv:1206.4522, 2012. URL http://arxiv.org/abs/1206.4522. version 1.

[43] Phillip I Good. *Resampling Methods: A Practical Guide to Data Analysis*. Birkhäuser Basel, 01 2006. ISBN 978-0-8176-4444-4], edition = 3rd. doi: 10.1007/0-8176-4444-X.

[44] Jian He, Enzo Veltri, Donatello Santoro, Guoliang Li, Giansalvatore Mecca, Paolo Papotti, and Nan Tang. Interactive and Deterministic Data Cleaning. In *Special Interest Group on Management of Data*, pages 893–907, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3531-7. doi: 10.1145/2882903.2915242. URL http://doi.acm.org/10.1145/2882903.2915242.

[45] Richard D Hipp. SQLite, 2020. URL https://www.sqlite.org/.

[46] Thomas Hofmann. Probabilistic latent semantic analysis. In *Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[47] Ihab F Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In *Special Interest Group on Management of Data*, pages 647–658, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007641. URL http://doi.acm.org/10.1145/1007568.1007641.

[48] Rezarta Islamaj Doğan, Donald C Comeau, Lana Yeganova, and W John Wilbur. Finding abbreviations in biomedical literature: three bioc-compatible modules and four bioc-formatted corpora. *Database: the journal of biological databases and curation*, 2014, 2014.

[49] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[50] Kayla Jacobs, Alon Itai, and Shuly Wintner. Acronyms: identification, expansion and disambiguation. *Annals of Mathematics and Artificial Intelligence*, 2018. ISSN 1573-7470. doi: 10.1007/s10472-018-9608-8. URL https://doi.org/10.1007/s10472-018-9608-8.

[51] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. ISBN 1461471370.

[52] Antonio J Jimeno-Yepes, Bridget T McInnes, and Alan R Aronson. Exploiting MeSH indexing in MEDLINE to generate a data set for word sense disambiguation. *BMC Bioinformatics*, 12(1):1–14, 2011. ISSN 1471-2105.

[53] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Conference on Human Factors in Computing Systems*, 2011. ISBN 9781450302289. doi: 10.1145/1978942.1979444.

[54] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage publications, 2018.

[55] Sanjay Krishnan, Daniel Haas, Michael J Franklin, and Eugene Wu. Towards Reliable Interactive Data Cleaning: A User Survey and Recommendations. In *Workshop on Human-In-the-Loop Data Analytics (Co-located with Special Interest Group on Management of Data)*, pages 9:1—-9:5, San Francisco, 2016. ACM. ISBN 978-1-4503-4207-0. doi: 10.1145/2939502.2939511. URL http://doi.acm.org/10.1145/2939502.2939511.

[56] Cheng-Ju Kuo, Maurice HT Ling, Woody Lin, and Chun-Nan Hsu. Bioadi: A machine learning approach to identifying abbreviations and definitions in biological literature. *BMC bioinformatics*, 10 Suppl 15:S7, 12 2009.

[57] John D Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

[58] Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, may 2014. URL http://arxiv.org/abs/1405.4053.

[59] Mong Li Lee, Tok Wang Ling, and Wai Lup Low. IntelliClean: A Knowledge-based Intelligent Data Cleaner. In *Knowledge Discovery and Data Mining*, pages 290–294, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347154. URL http://doi.acm.org/10.1145/347090.347154.

[60] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[61] Chao Li, Lei Ji, and Jun Yan. Acronym Disambiguation Using Word Embedding. In *Conference on Artificial Intelligence*, pages 4178–4179. AAAI, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2888116.2888316.

[62] Jianhua Lin. Divergence measures based on the Shannon entropy. *Transactions on Information theory*, 37(1):145–151, 1991.

[63] Jie Liu, Caihua Liu, and Yalou Huang. Multi-granularity sequence labeling model for acronym expansion identification. *Information Sciences*, 378:462–474, feb 2017. ISSN 00200255. doi: 10.1016/j.ins.2016.06.045. URL https://www.sciencedirect.com/science/article/pii/S0020025516304741.

[64] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Computing Research Repository*, arXiv:1907.11692, 2019. URL https://arxiv.org/abs/1907.11692. version 1.

[65] Steve Lohr. The age of Big Data. New York Times, 2012. ISSN 0036-8075. URL https://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html.

[66] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[67] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers. Big Data: The Next Frontier for Innovation, Competition, and Productivity. Technical Report June, McKinsey Global Institute, 2011.

[68] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository*, arXiv:1301.3781, 2013. URL https://arxiv.org/abs/1301.3781. version 3.

[69] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

[70] Tova Milo and Sagit Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Very Large Data Bases*, pages 122–133, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-566-5. URL http://dl.acm.org/citation.cfm?id=645924.671326.

[71] L Morency, A Quattoni, and T Darrell. Latent-Dynamic Discriminative Models for Continuous Gesture Recognition. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. doi: 10.1109/CVPR.2007.383299.

[72] Mashaal Musleh, Mourad Ouzzani, Nan Tang, and AnHai Doan. CoClean: Collaborative Data Cleaning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 2757–2760, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356. doi: 10.1145/3318464.3384698. URL https://doi.org/10.1145/3318464.3384698.

[73] Paulo Oliveira, Fátima Rodrigues, and Helena Galhardas. A Taxonomy of Data Quality Problems. In *Workshop on Data and Information Quality*, 2005.

[74] Cátia Borges Ormonde. CLEENEX: Iterative Data Cleaning with User Intervention, 2017.

[75] Youngja Park and Roy J Byrd. Hybrid Text Mining for Finding Abbreviations and their Definitions. In *Empirical Methods in Natural Language Processing*, 2001.

[76] Rebecca Passonneau. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy, May 2006. European Language Resources Association (ELRA).

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[78] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[79] João L. M. Pereira and Helena Galhardas. Approximate Duplicate Elimination using State-Of-The-Art Tools: a Comparison. In *INFORUM*, pages 99–110, Aveiro, Portugal, 2017.

[80] João L M Pereira, Helena Galhardas, and Dennis Shasha. Acronym Expander at SDU@ AAAI-21: an Acronym Disambiguation Module. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org, 2021. URL http://ceur-ws.org/Vol-2831/paper32.pdf.

[81] João L. M. Pereira, João Casanova, Helena Galhardas, and Dennis Shasha. AcX: system, techniques, and experiments for Acronym eXpansion. *Proceedings of the VLDB Endowment*, 15(11): 2530–2544, 2022. URL https://vldb.org/pvldb/vol15/p2530-pereira.pdf.

[82] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. *DORA THE EXPLORER: Exploring Very Large Data With Interactive Deep Reinforcement Learning*, pages 4769–4773. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450384469. URL https://doi.org/10.1145/3459637.3481967.

[83] Tomas Petricek, Gerrit J.J. van den Burg, Alfredo Nazábal, Taha Ceritli, Ernesto Jiménez-Ruiz, and Christopher K. I. Williams. Ai assistants: A framework for semi-automated data wrangling. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–12, 2022. doi: 10.1109/TKDE.2022.3222538.

[84] Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation. In *International Conference on Computational Linguistics*, pages 3285–3301, 2021. doi: 10.18653/v1/2020.coling-main.292.

[85] Roman Prokofyev, Gianluca Demartini, Alexey Boyarsky, Oleg Ruchayskiy, and Philippe Cudré-Mauroux. Ontology-Based Word Sense Disambiguation for Scientific Literature. In Pavel Serdyukov, Pavel Braslavski, Sergei O Kuznetsov, Jaap Kamps, Stefan Rüger, Eugene Agichtein, Ilya Segalovich, and Emine Yilmaz, editors, *Advances in Information Retrieval*, pages 594–605, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-36973-5.

[86] J Pustejovsky, J Castaño, B Cochran, M Kotecki, and M Morrell. Automatic extraction of acronym-meaning pairs from MEDLINE databases. *Studies in Health Technology and Informatics*, 84(Pt 1):371–375, 2001. ISSN 0926-9630 (Print).

[87] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

[88] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, Long Papers, pages 99–110, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1010.

[89] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[90] Protiva Rahman, Courtney Hebert, and Arnab Nandi. ICARUS: Minimizing Human Effort in Iterative Data Completion. *Proceedings of the Very Large Data Bases Endowment*, 11(13):2263–2276, 2018. ISSN 2150-8097. doi: 10.14778/3275366.3284970. URL https://doi.org/10.14778/3275366.3284970.

[91] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Empirical Methods in Natural Language Processing*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[92] El Kindi Rezig, Mourad Ouzzani, Ahmed K Elmagarmid, Walid G Aref, and Michael Stonebraker. Towards an End-to-End Human-Centric Data Cleaning Framework. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA'19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367912. doi: 10.1145/3328519.3329133. URL https://doi.org/10.1145/3328519.3329133.

[93] Xin Rong. word2vec Parameter Learning Explained. *Computing Research Repository*, arXiv:1411.2, 2014. URL http://arxiv.org/abs/1411.2738. version 4.

[94] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 3rd edition*. Prentice Hall, 2009. ISBN 0136042597. doi: 10.1017/S0269888900007724.

[95] Barna Saha and Divesh Srivastava. Data quality: The other face of Big Data. In *International Conference on Data Engineering*, pages 1294–1297. IEEE, 2014.

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

[97] Ariel S Schwartz and Marti A Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, pages 451–462, 2002.

[98] Utpal Kumar Sikdar and Björn Gambäck. A Feature-based Ensemble Approach to Recognition of Emerging and Rare Named Entities. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 177–181, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4424. URL https://aclanthology.org/W17-4424.

[99] Aadarsh Singh and Priyanshu Kumar. SciDr at SDU-2020: IDEAS–Identifying and Disambiguating Everyday Acronyms for Scientific Domain. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org, 2021. URL http://ceur-ws.org/Vol-2831/paper31.pdf.

[100] Sunghwan Sohn, Donald C Comeau, Won Kim, and W John Wilbur. Abbreviation definition identification based on automatic precision estimates. *BMC bioinformatics*, 9(1):402, 2008.

[101] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for Language Understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

[102] Karen Sparck Jones. Document Retrieval Systems. chapter A Statisti, pages 132–142. Taylor Graham Publishing, London, UK, UK, 1988. ISBN 0-947568-21-2. URL http://dl.acm.org/citation.cfm?id=106765.106782.

[103] Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[104] Ji Sun, Dong Deng, Ihab F Ilyas, Guoliang Li, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Technical Report: Optimizing Human Involvement for Entity Matching and Consolidation. *Computing Research Repository*, arXiv:1906.0, 2019. URL http://arxiv.org/abs/1906.06574. version 1.

[105] Susan Moore. How to Create a Business Case for Data Quality Improvement. Gartner, 2018. URL https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality-improvement/.

[106] Nikki Swartz. Gartner warns firms of 'dirty data'. *Information Management Journal*, 41(3):6, 2007.

[107] Aditya Thakker, Suhail Barot, and Sudhir Bagul. Acronym Disambiguation: A Domain Independent Approach. *Computing Research Repository*, arXiv:1711.09271, 2017. URL https://arxiv.org/abs/1711.09271. version 3.

[108] Saravanan Thirumuruganathan, Laure Berti-Equille, Mourad Ouzzani, Jorge-Arnulfo Quiane-Ruiz, and Nan Tang. UGuide: User-Guided Discovery of FD-Detectable Errors. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1385–1397, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450341974. doi: 10.1145/3035918.3064024. URL https://doi.org/10.1145/3035918.3064024.

[109] Tiago Francisco. CLEENEX - Debugger, 2015.

[110] Amir Pouran Ben Veyseh, Franck Dernoncourt, Walter Chang, and Thien Huu Nguyen. MadDog: A Web-based System for Acronym Identification and Disambiguation. In *European Chapter of the Association for Computational Linguistics*, 2021.

[111] Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, Walter Chang, and Leo Anthony Celi. Acronym Identification and Disambiguation Shared Tasks for Scientific Document Understanding. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org, 2021. URL http://ceur-ws.org/Vol-2831/paper33.pdf.

[112] Maksims Volkovs, Fei Chiang, Jaroslaw Szlichta, and Renée J Miller. Continuous data cleaning. In *International Conference on Data Engineering*, pages 244–255. IEEE, 2014.

[113] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. CrowdER: Crowdsourcing Entity Resolution. *Proceedings of the VLDB Endowment*, 5(11), 2012.

[114] Mohamed Yakout, Ahmed K Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F Ilyas. Guided data repair. *Proceedings of the Very Large Data Bases Endowment*, 4(5):279–289, 2011. doi: 10.14778/1952376.1952378.

[115] Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. Global entity disambiguation with bert. *Computing Research Repository*, arXiv:1909.00426, 2019. URL `https://arxiv.org/abs/1909.00426`. version 3.

[116] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G Carbonell, Ruslan Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Advances in Neural Information Processing Systems*, 12, 2019. URL `http://arxiv.org/abs/1906.08237`.

[117] Anna Yarygina and Natalia Vassilieva. High-recall Extraction of Acronym-definition Pairs with Relevance Feedback. In *Joint Extending Database Technology and International Conference on Database Theory Workshops*, pages 21–28, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1143-4. doi: 10.1145/2320765.2320781. URL `http://doi.acm.org/10.1145/2320765.2320781`.

Appendices

Appendix A

Acronym eXpansion (AcX) Dataset Creation and Structures Details

A.1 User-Generated Dataset Creation Process

To further the annotation process, we created two forms: (i) that promoted the annotation task and allowed for students to register[1] in this task and (ii) used for the annotation process itself[2]. The first form explained the overall annotation task and stated the prize each student annotator could win. Immediately after submitting form (i), an e-mail was sent to each student containing the English Wikipedia documents to annotate with a link to form (ii). Furthermore, in form (ii) additional details, instructions, and examples of the annotation process were given to each student explaining the annotation process.

A.2 Data Structures Used in AcX

Each dataset referenced in Section 4.2.1.1 and Section 4.3.1.1 were in different formats and used a different annotation notation for the acronym-expansion pairs in the dataset documents. To facilitate access to each dataset by the benchmark, we used the following dictionaries to implement the AcX database:

- A dictionary that maps each document id to the corresponding raw text and for out-expansion an additional dictionary is provided with the preprocessed text.
- A dictionary that maps each document id to the acronym-expansion pairs whose acronyms are present in text.
- A dictionary that maps each acronym in the corpus to the corresponding in- and/or out- expansions with the document ids where they appear.

[1] https://forms.gle/hWJR2K64XzjpYrYY9
[2] https://forms.gle/VH1SCf2nr1PBAZK18

Appendix B

In-expansion Glossary-level Results

We report Glossary-level Precision, Recall, and F1-measure values for the biomedical datasets (i.e., Medstract, Schwartz and Hearst, BIOADI and Ab3P) for acronym extraction in Table B.1 and for acronym-expansion pair extraction in Table B.2. We report Glossary-level Precision, Recall, and F1-measure values for botch acronym and pair for SciAI dataset in Table B.3 and for User-Generated dataset in Table B.4.

Acronym and In-expansion Technique	**Acronym**														
	Medstract			**SH**			**BIOADI**			**Ab3P**			**Average**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
SH	**100.00%**	**88.63%**	**93.97%**	**99.48%**	**79.83%**	**88.58%**	**98.69%**	**78.26%**	**87.29%**	98.20%	**77.84%**	**86.84%**	**99.09%**	**81.14%**	**89.17%**
MadDog	**100.00%**	63.63%	77.78%	95.96%	48.97%	64.85%	97.76%	54.24%	69.77%	**99.13%**	64.77%	78.35%	98.21%	57.90%	72.69%
SciBERT	80.64%	56.81%	66.67%	81.46%	68.72%	74.55%	85.51%	74.53%	79.64%	85.18%	71.87%	77.96%	83.20%	67.98%	74.71%
SciBERT with External Data	86.48%	72.72%	79.01%	84.47%	76.13%	80.08%	88.78%	78.67%	83.42%	87.01%	76.13%	81.21%	86.69%	75.91%	80.93%
SciDr	93.10%	61.36%	73.97%	84.30%	59.67%	69.87%	90.40%	64.38%	75.21%	88.56%	68.18%	77.04%	89.09%	63.40%	74.02%
SciDr-in with External Data	91.89%	77.27%	83.95%	91.74%	77.77%	84.18%	92.10%	79.71%	85.46%	88.11%	71.59%	78.99%	90.96%	76.59%	83.15%

Table B.1: In-expansion techniques Glossary-level Precision, Recall and F1-measures for acronym for each biomedical dataset (Medstract, SH, BIOADI, and Ab3p) and the averages.

Acronym and In-expansion Technique	**Pair**														
	Medstract			**SH**			**BIOADI**			**Ab3P**			**Average**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
SH	**100.00%**	**88.63%**	**93.97%**	**95.38%**	**76.54%**	**84.93%**	**93.99%**	**74.53%**	**83.14%**	94.98%	**75.28%**	**83.99%**	**96.09%**	**78.75%**	**86.51%**
MadDog	92.85%	59.09%	72.22%	91.93%	46.91%	62.12%	87.31%	48.44%	62.31%	**95.21%**	62.21%	75.25%	91.83%	54.16%	67.98%
SciBERT	64.51%	45.45%	53.33%	69.26%	58.43%	63.39%	66.27%	57.76%	61.72%	74.41%	62.78%	68.10%	68.61%	56.11%	61.64%
SciBERT with External Data	75.67%	63.63%	69.13%	74.42%	67.07%	70.56%	72.89%	64.59%	68.49%	76.29%	66.76%	71.21%	74.82%	65.51%	69.85%
SciDr	79.31%	52.27%	63.01%	71.51%	50.61%	59.27%	75.29%	53.62%	62.63%	80.81%	62.21%	70.30%	76.73%	54.68%	63.80%
SciDr-in with External Data	91.89%	77.27%	83.95%	81.55%	69.13%	74.83%	85.16%	73.70%	79.02%	79.72%	64.77%	71.47%	84.58%	71.22%	77.32%

Table B.2: In-expansion techniques Glossary-level Precision, Recall and F1-measures for pair for each biomedical dataset (Medstract, SH, BIOADI, and Ab3p) and the averages.

Acronym and In-expansion Technique	**SciAI**					
	Acronym			**Pair**		
	P	**R**	**F1**	**P**	**R**	**F1**
SH	94.79%	82.63%	88.29%	91.06%	79.38%	84.82%
MadDog-in	**98.43%**	85.97%	91.78%	**96.37%**	84.17%	89.86%
SciBERT	94.63%	**93.58%**	94.10%	90.57%	**89.56%**	90.06%
SciBERT with External data	95.29%	93.49%	**94.38%**	91.10%	89.39%	90.24%
SciDr-in	96.75%	91.78%	94.20%	93.41%	88.62%	**90.95%**
SciDr-in with External data	96.91%	91.36%	94.05%	92.46%	87.16%	89.74%

Table B.3: In-expansion techniques Glossary-level Precision, Recall and F1-measures for acronym and pair extraction and for the SciAI dataset.

Acronym and In-expansion Technique	User-Generated					
	Acronym			Pair		
	P	R	F1	P	R	F1
SH	90.42%	**70.83%**	**79.43%**	85.10%	**66.67%**	74.76%
MadDog-in	**92.22%**	69.17%	79.04%	**87.78%**	65.83%	**75.23%**
SciBERT	64.13%	49.16%	55.66%	56.52%	43.33%	49.05%
SciBERT with External data	48.63%	59.17%	53.38%	44.52%	54.17%	48.87%
SciDr-in	76.67%	57.50%	65.71%	67.78%	50.83%	58.09%
SciDr-in with External data	85.54%	59.17%	69.95%	80.72%	55.83%	66.01%

Table B.4: In-expansion techniques Glossary-level Precision, Recall and F1-measures for acronym and pair extraction for the User Generated dataset.