ADVANCED TEXT PROCESSING WORKFLOWS IN A WEB-BASED TEXT MINING PLATFORM

Matic Perovšek

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Matic Perovšek

ADVANCED TEXT PROCESSING WORKFLOWS IN A WEB-BASED TEXT MINING PLATFORM

Doctoral Dissertation

NAPREDNI DELOTOKI ZA PROCESIRANJE TEKSTOV V SPLETNI PLATFORMI ZA TEKSTOVNO RUDARJENJE

Doktorska disertacija

Supervisor: Prof. Dr. Bojan Cestnik

Co-Supervisor: Prof. Dr. Nada Lavrač

Ljubljana, Slovenia, March 2016

Acknowledgments

This thesis would never have been finished without the help of a number of people. Their support and help throughout my studies is greatly appreciated.

First of all I would like to thank my supervisors Bojan Cestnik and Nada Lavrač for their infinite amount of patience, enthusiasm, support and their always helpful and insightful research ideas and comments. My thanks also go to the members of the evaluation board Tomaž Erjavec, Igor Mozetič and Ljupčo Todorovski for comments and suggestions, which helped me to improve the quality of this thesis.

I also wish to thank the funding bodies that financially supported my research: the Slovenian Research Agency, the Department of Knowledge Technologies at the Jožef Stefan Institute, the Jožef Stefan International Postgraduate School, and the European Commission for funding the research projects WHIM, ConCreTe and MUSE in which I was involved.

Thanks to all the colleagues at the Department of Knowledge Technologies at the Jožef Stefan Institute for creating a great working environment. For help with many theoretical and engineering problems I would like to thank Anže Vavpetič, Janez Kranjc and Jan Kralj. Furthermore, I would also like to thank Darko Aleksovski and Vid Podpečan for always having time for a friendly talk.

Special thanks go to Matjaž Juršič and Miha Grčar for their selfless supply of inspiring ideas, programming knowledge, and their valuable time for countless very productive and insightful discussions that greatly motivated my research work.

Last but not least, I must thank my family and friends for their unending encouragement and support. Finally, I am most grateful to my dearest Didka for her love, understanding, support and for believing in me even in the most difficult times.

Abstract

Text mining is a field of data mining, drawing its techniques from machine learning, natural language processing, information retrieval, information extraction and knowledge management. While text mining is a mature area of research, with numerous open source algorithms and NLP software libraries available—such as NLTK and scikit-learn—text mining and natural language processing (NLP) experiments are still difficult to reproduce, including the difficulty of systematic comparison of algorithms. To this end, a number of attempts have been made to develop easy-to-use workflow management systems, allowing users to compose complex processing pipelines in a visual programming manner.

The thesis addresses several open questions and technology gaps in text mining (TM) and natural language processing (NLP), following three main research directions: (a) development of advanced TM and NLP infrastructures, (b) development of advanced workflows for natural language processing and bisociative knowledge discovery, and (c) development of advanced workflows for transforming relational data mining into text mining problems.

We developed a web-based text mining platform TextFlows, which facilitates the construction and execution of text mining and NLP workflows. The platform serves as an easy-to-access integration platform for the advanced text mining and NLP approaches presented in this thesis. A survey of existing libraries and workflow management systems for text mining is provided, outlining their abilities and deficiencies. Moreover, the dissertation addresses several novel knowledge discovery scenarios, for which it proposes the methodology and presents the developed reference implementation, together with their evaluation on real-world text mining and data mining tasks. In particular, the thesis focuses on the development of advanced workflows for natural language processing, advanced workflows for bisociative knowledge discovery, and advanced workflows for transforming relational data mining problems into text mining problems. The thesis also provides reference implementations for solving the addressed scenarios in the developed TextFlows text mining platform, which allows for workflow sharing and experiment repeatability.

Povzetek

Tekstovno rudarjenje je področje podatkovnega rudarjenja, ki temelji na tehnikah strojnega učenja, obdelave naravnega jezika, ekstrakcije informacij, informacijskega poizvedovanja in upravljanja znanja. Tekstovno rudarjenje je zrelo področje raziskovanja, ki ponuja uporabniku številne odprtokodne algoritme in knjižnice za obdelavo naravnega jezika, kot sta npr. knjižnici NLTK in scikit-learn. Kljub temu pa so tako eksperimenti tekstovnega rudarjenja in obdelave naravnega jezika kot tudi sistematična primerjava algoritmov še vedno težko izvedljivi. V ta namen je bilo opravljenih več (le deloma uspešnih) poizkusov razvoja sistemov, ki bi uporabniku z enostavno uporabo in načinom vizualnega programiranja omogočili sestavo in izvajanje kompleksnih postopkov za obdelavo tekstovnih podatkov.

Doktorska disertacija obravnava več odprtih vprašanj in tehnoloških vrzeli tekstovnega rudarjenja ter obdelave naravnega jezika v okviru treh glavnih smeri: (a) razvoj naprednih infrastruktur za tekstovno rudarjenje in obdelavo naravnega jezika, (b) razvoj naprednih delotokov za odkrivanje meddomenskih zakonitosti v podatkih, (c) razvoj naprednih delotokov za pretvorbo problema relacijskega rudarjenja podatkov na problem tekstovnega rudarjenja.

Razvili smo odprtokodno oblačno platformo TextFlows za tekstovno rudarjenje podatkov, ki omogoča gradnjo in izvajanje delotokov tekstovnega rudarjenja podatkov in obdelave naravnega jezika. Platforma hrati omogoča tudi integracijo ter enostaven dostop do naprednih delotokov, razvitih v okviru te doktorske disertacije. Disertacija vključuje tudi pregled obstoječih knjižnic in sistemov za rudarjenje tekstovnih podatkov ter povzema njihove zmožnosti in pomanjkljivosti. Obravnavanih je tudi več novih scenarijev odkrivanja znanja, za katere smo razvili novo metodologijo in implementacijo, hkrati pa podamo tudi evaluacijo na problemih tekstovnega in podatkovnega rudarjenja. Disertacija se osredotoča na razvoj naprednih delotokov za obdelavo naravnega jezika, naprednih delotokov za odkrivanje meddomenskih zakonitosti v podatkih in delotokov, ki preoblikujejo problem relacijskega podatkovnega rudarjenja v problem tekstovnega rudarjenja. Implementacije postopkov za reševanje obravnavanih scenarijev so na voljo v platformi TextFlows, kar omogoča enostavno izmenjavo razvitih delotokov in ponovljivost eksperimentov.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter places our work into a broader context of data mining, provides the line of reasoning for choosing the given research topic and states the contributions to science of this thesis. First we provide an outline of the topic addressed, followed by the purpose of the dissertation, its goals and scientific contributions. We conclude with a short structural overview of the rest of the thesis.

1.1 Problem Description

Text mining [1] is a research area that deals with the construction of models and patterns from text resources, aiming at solving tasks such as text categorization and clustering, taxonomy construction, and sentiment analysis. This research area, also known as text data mining or text analytics, is usually considered as a sub-field of data mining (DM) research [2], but can be viewed also more generally as a multidisciplinary field drawing its techniques from data mining, machine learning, natural language processing (NLP) [3], [4], information retrieval (IR) [5], information extraction (IE) [6] and knowledge management [7]. From a procedural point of view, text mining processes typically follow the CRISP-DM reference process model for data mining [8], which proposes six phases when working on a DM project: problem understanding, data understanding, data preparation, modeling, evaluation, and deployment. Text mining can be distinguished from general data mining by special procedures applied in the data preparation phase, where unstructured or poorly structured text needs to be converted into organized data, structured as a table of instances (rows) described by attributes (columns). In the modeling phase, such a table of instances is used by the standard or slightly adapted data mining algorithms to uncover interesting information hidden in the data. Two typical approaches are using clustering algorithms to find groups of similar instances, and learning classifiers to categorize new instances.

While text mining is a mature area of research, with numerous open source algorithms and NLP software libraries available—such as LATINO[1] [9], NLTK[2] [10] and scikit-learn[3] [11]—text mining and NLP experiments are still difficult to reproduce, including the difficulty of systematic comparison of algorithms. To this end, a number of attempts have been made to develop easy-to-use workflow management systems, allowing users to compose complex processing pipelines in a visual programming manner. These attempts are presented in Chapter 2.

[1]LATINO (Link Analysis and Text Mining Toolbox) is open-source—mostly under the LGPL license—and is available at `https://github.com/LatinoLib/LATINO`

[2]`http://www.nltk.org`

[3]`http://scikit-learn.org`

The thesis addresses several open questions and technology gaps in text mining (TM) and natural language processing (NLP), following three main research directions: (a) development of advanced TM and NLP infrastructures, (b) development of advanced workflows for natural language processing and bisociative knowledge discovery, and (c) development of advanced workflows for transforming relational data mining problems into text mining problems. The problem description below follows these three research directions.

1.2 Purpose of the Dissertation

The purpose of this dissertation is multifold. First, we facilitate the construction and execution of text mining and NLP workflows by developing a web-based text mining platform. The developed TextFlows platform, described in this thesis, is a new open source, web-based text mining platform that supports the design and composition of scientific procedures implemented as executable workflows. As a fork of ClowdFlows [12], TextFlows has inherited its service-oriented architecture that allows the user to utilize arbitrary web services as workflow components. TextFlows is oriented towards text analytics and offers a number of algorithms for text mining and natural language processing. The platform is implemented as a cloud-based web application, aiming to overcome various deficiencies of similar text analytics platforms by providing novel features that should be beneficial to the text mining community. In contrast to existing text analytics workflow management systems, the developed platform is the only one with all the following properties. It is simple (i.e. enables visual programming, is web-based and requires no installation), enables workflow sharing and reuse, and is open source. Moreover, the platform enables combining workflow components (also called ‘widgets’) from different contexts (e.g., using clustering in relational data mining) and from different software sources (e.g., building ensemble classifiers from different libraries). To do so, it provides a unified input-output representation, which enables interoperability between widget libraries through automated data type conversion. It uses a common text representation structure and advocates the development of ‘hubs’ for algorithm execution. The platform serves as an easy-to-access integration platform for the advanced text mining and NLP approaches presented in this thesis. Furthermore, a survey of existing libraries and workflow management systems for text mining is provided, outlining their abilities and deficiencies. Moreover, the dissertation addresses several novel knowledge discovery scenarios for which it proposes the methodology and the developed reference implementations, together with their evaluation on real-world text mining and data mining tasks. In particular, the thesis focuses on the development of advanced workflows for natural language processing, advanced workflows for bisociative knowledge discovery, and advanced workflows for transforming relational data mining problems into text mining problems. The thesis also provides reference implementations for solving the addressed scenarios in the developed TextFlows text mining platform, which allows for workflow sharing and experiment repeatability.

1.3 Hypotheses and Goals

The main hypothesis of the thesis is that textual data can be efficiently processed with a system that implements the visual programming paradigm and utilizes computing by means of distributed hardware and software resources to improve scalability and adaptation to data of large proportions, which outperforms existing workflow management systems. By unifying functionalities of different knowledge discovery platforms and adding features to facilitate text mining joined with natural language processing, we should be able to address novel scenarios which could not be considered within the same platform until now, such as comparison of stemmers and Part-of-Speech taggers from different libraries, literature-based discovery in a text mining platform, and relational data mining empowered by a text mining inspired technique.

Another hypothesis investigated in this thesis is that by using advanced text mining approaches we can extract the hidden and till now uncovered links and patterns across different domains of investigation. These cross-domain links, named bisociations [13], [14], can show us, for example, a solution to a problem in one domain, which can be applied in another domain, but was not yet explored by the experts because of their high specialization on their own domain of expertise. Discovered bisociations may also trigger new ideas which could lead to new discoveries in a given domain. The assumption is that bisociative links should provide novel insights into the problem domain and one has to cross contexts in order to find them.

We also investigate the hypothesis that by using text mining approaches we can develop a propositionalization technique, which can be seen as a transformation of a relational database into a corpus of text documents. The technique should transform relational data into a single table format by constructing simple, easy to understand features, acting as words in the transformed Bag-of-Words representation, resulting also in improved scalability in order to handle large datasets.

The main goal of the dissertation is to provide a methodology and reference implementation of a cloud-based text mining and natural language processing platform, to meet the requirements of mining ever-growing amounts of textual data. This reference platform integrates:

- a visual programming paradigm with a graphical user interface accessible from a web browser on a computer or mobile device,
- a framework for sharing data and workflows and making them publicly available to a broad audience,
- a cloud-based architecture that allows the distributed text processing on a cluster of machines, and
- an implementation of numerous open source algorithms and available NLP software libraries to allow for easy reproduction of text mining and NLP experiments.

The dissertation also addresses novel knowledge discovery scenario to demonstrate the usefulness of the cloud-based text mining methodology and the practical utility of the implemented reference platform. The goal of the dissertation is to develop methodologies composed from several text mining and natural language processing components. These scenarios include:

- a natural language processing use cases featuring implementations and comparisons of methods from several knowledge discovery libraries in a unified workflow environment,
- a complex literature-based discovery scenario featuring various text mining components (tokenization, stemming, lemmatization, document visualization, bisociative links discovery), and
- a relational data mining scenario where a text mining inspired approach is evaluated in comparison with existing propositionalization approaches on several relational datasets.

1.4 Scientific Contributions

The main scientific contributions of the thesis are as follows:

- A web-based text mining platform, named TextFlows, enabling advanced workflow construction, execution, modification and sharing:
  - Compared to existing text mining workflow management systems, the developed platform is the only one with the following three properties: simplicity (it is web based and requires no installation), enables workflow sharing, and is open source.
  - The platform enables combining algorithms from different contexts (e.g., using clustering in relational data mining) and from different software sources (e.g., building ensemble classifiers from different libraries).
  - Unified input-output representation enables interoperability between widget libraries through automated data type conversion.
- Advanced bisociative literature mining techniques:
  - We developed advanced bisociative knowledge discovery workflows, including a banded matrix approach to bisociative bridging term discovery and improved the identification of bridging terms by using synonyms and controlled dictionaries.
  - Evaluation of a banded matrix approach when used as a heuristic for bisociative cross-domain knowledge discovery.
  - Application of bisociative knowledge discovery in a narrative ideation context.
- Novel propositionalization technique called wordification:
  - The proposed approach has the advantage of producing easy to understand hypotheses, using much simpler features than the comparable propositionalization systems.
  - The experimental comparison with other propositionalization techniques showing that the proposed wordification approach using the J48 and LibSVM classifiers performs favorably in terms of accuracy and efficiency, compared to the state-of-the-art propositionalization algorithms.
  - It is shown that the presented approach has the capacity to solve non-standard relational data mining tasks, such as clustering on relational databases, association rule learning, outlier detection, etc.
  - The implementation of the wordification technique as a reusable, online-available experimental workflow (from connecting to a relational database management server to visualizing the experimental results and evaluation), which enables researchers to reuse the developed software in their own experiments.

The scientific contributions of this work were published in the following papers:

Text mining and natural language processing

M. Perovšek, V. Podpečan, J. Kranjc, T. Erjavec, S. Pollak, N. Q. Do Thi, X. Liu, C. Smith, M. Cavazza, and N. Lavrač, "Text mining platform for NLP workflow design, replication and reuse," in *Proceedings of IJCAI Workshop on Replicability and Reusability in Natural Language Processing: From Data to Software Sharing*, 2015. [Online]. Available: `https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhZGFwdGl2ZW5scDIwMTV8Z3g6NTU1M2Q5MmYzZTYyNTZkNQ`.

M. Perovšek, J. Kranjc, T. Erjavec, B. Cestnik, and N. Lavrač, "TextFlows: A visual programming platform for text mining and natural language processing," *Science of Computer Programming: Special Issue on Knowledge-based Software Engineering*, vol. 121, pp. 128–152, 2016.

Literature-based cross-domain knowledge discovery

M. Perovšek, B. Cestnik, T. Urbančič, S. Colton, and N. Lavrač, "Towards narrative ideation via cross-context link discovery using banded matrices," in *Proceedings of the 12th International Conference on Intelligent Data Analysis*, Springer, 2013, pp. 333–344.

M. T. Llano, R. Hepworth, S. Colton, J. Gow, J. Charnley, N. Lavrač, M. Žnidaršič, M. Perovšek, M. Granroth-Wilding, and S. Clark, "Baseline methods for automated fictional ideation," in *Proceedings of the International Conference on Computational Creativity*, 2014, pp. 211–219.

Relational data mining

M. Perovšek, A. Vavpetič, and N. Lavrač, "A wordification approach to relational data mining: Early results," in *Late Breaking Papers of the 22nd International Conference on Inductive Logic Programming*, 2012, pp. 56–61. [Online]. Available: `http://ceur-ws.org/Vol-975/paper-06.pdf`.

M. Perovšek, A. Vavpetič, B. Cestnik, and N. Lavrač, "A wordification approach to relational data mining," in *Proceedings of the 16th International Conference, Discovery Science 2013*, Springer, 2013, pp. 141–154.

N. Lavrač, M. Perovšek, and A. Vavpetič, "Propositionalization online," in *Proceedings of the European Conference, ECML PKDD 2014, Nancy, France*, Springer, 2014, pp. 456–459.

M. Perovšek, A. Vavpetič, J. Kranjc, B. Cestnik, and N. Lavrač, "Wordification: Propositionalization by unfolding relational data into bags of words," *Expert Systems with Applications*, vol. 42, no. 17, pp. 6442–6456, 2015.

1.5 Thesis Structure

In Chapter 1 we have already presented the grounds for this thesis by describing the main motivation and purpose, as well as providing the hypothesis goals and giving a summary of contributions. This section provides the overview of the thesis.

The related work to this thesis is presented in Chapter 2, where we first place our work into a broad context of text mining. Afterwards, existing platforms, libraries, tools and environments for text mining and natural language detection are evaluated.

We present the motivation for developing a new cloud-based text mining platform, named TextFlows, in Chapter 3. Moreover, we also present the technical background and architecture of the TextFlows platform, along with its key text mining components. The architecture of the system is presented in detail along with specific data structures that allow efficient text mining in a workflow environment. We also describe the concept of workflows, their implementation, execution and sharing, followed by presenting the widget repository and the implemented modules of the platform. In Chapter 3 we also explain the relationship between TextFlows and ClowdFlows, while also comparing TextFlows with other state-of-the-art workflow management systems along several dimensions that affect the usefulness of each system, where we identified their distinguishing features, abilities, and deficiencies.

Chapter 4 addresses novel natural language processing and knowledge discovery scenarios which rely on the TextFlows implementation, described in Chapter 3. The main goal of this chapter is to ultimately justify the development of a web-based, cloud-based knowledge discovery platform for text mining and NLP processing. For the natural language processing scenario we demonstrate the advanced features of the developed platform on five use cases. First, we propose a document acquisition and text preprocessing workflow, which is also used as a part of the workflow in the remaining four use cases. Second, we also describe the Kenyan elections dataset [15], [16], which is the dataset used in the experiments and proposed workflows. Third, we propose a workflow for comparing different classifiers from different libraries for a text classification problem, as well as a comparison of several stemmers/lemmatizers on the same text categorization problem. Moreover, we also show a comparison of Part-of-Speech taggers from different software libraries. Finally, we present an approach for outlier detection on document corpora as a workflow implemented in TextFlows.

The literature-based cross-domain knowledge discovery scenario [14] is presented in Chapter 5. First, we illustrate the problem of bridging term ranking followed by a brief description of CrossBee [17]–[19]. Next, we provide an overview of the CrossBee methodology, as well as the elementary and ensemble heuristics used in its bridging term discovery process. We also show the implementation of the complex CrossBee methodology as a workflow in TextFlows, where the platform acts as the enabling technology for implementing the developed cross-domain link discovery approach. Furthermore, we study a new type of elementary heuristics for bridging term ranking, which use banded matrices to discover structures which reveal the relations between the rows (representing documents) and columns (representing words/terms) of a given data matrix (representing a set of documents). Moreover, we also propose an extension of the CrossBee methodology, which incorporates background knowledge—controlled vocabularies—into the bridging term discovery process. Finally, we evaluate the developed framework on multiple identified textual sources and comparing results to previously published results in order to verify the importance of the retrieved bisociations.

In Chapter 6 we address a relational data mining (RDM) and inductive logic programming (ILP) scenario [20]–[23] by proposing a new methodology, called *wordification*, for data propositionalization [24], [25]. Wordification constructs simple, easy to understand features, acting as words in the transformed Bag-of-Words representation [1]. The main advantages of the approach are: simple implementation, accuracy comparable to competitive methods, and greater scalability, as it performs several times faster on all experimental databases. In Chapter 6 we describe the wordification methodology in detail and the implementation of the evaluation procedure as an executable workflow in the web-based text mining platform TextFlows. The implemented workflows include also several other ILP and RDM algorithms as well as the utility components that were added to the platform to enable access to these techniques to a wider research audience.

Chapter 7 concludes the thesis by presenting a summary with respect to the hypothesis and goals presented in Chapter 1. Finally, we also present several ideas for further work.

Chapter 2

Background and Related Work

This chapter introduces the background of this thesis and describes the related work. First, a description of the area of text mining is provided, followed by an evaluation of existing platforms, libraries, tools and environments for text mining and natural language processing. Secondly, we introduce a specific type of information extraction task referred to as bisociation discovery and in particular bisociative literature mining, which is the focus of a part of this thesis. Finally, we present the area of relational data mining (RDM) and propositionalization.

2.1 Text Mining and Natural Language Processing

Text mining [1] is the process of discovering of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking of the extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation. Text mining can be viewed also more generally as a multidisciplinary field drawing its techniques from data mining, machine learning, natural language processing (NLP) [3], [4], information retrieval (IR) [5], information extraction (IE) [6] and knowledge management [7].

Text mining research aims at solving tasks such as text categorization and clustering, taxonomy construction, and sentiment analysis. This research area, also known as text data mining or text analytics, is usually considered as a subfield of data mining (DM) research [2]. The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text rather than from structured databases of facts. Databases are designed for programs to process automatically; text is written for people to read. However, there is a field called natural language processing [3], [4] which is making a lot of progress in doing small subtasks in text analysis. Natural language processing (NLP) refers to the computer processing of natural languages, for some purpose, regardless of the processing depth. The so-called 'natural language' stands for the languages we use in our daily life, such as English, Russian, Chinese, it is synonymous with human language, mainly to be distinguished from formal language, including programming languages. As it stands, natural language is the most natural and most common form of human communication, not only in the spoken form, but also the written language is growing exponentially in recent years, when the mobile Internet is hot with all the social media. Compared with the formal language, natural language is much more complex, often with omissions and ambiguity, making it difficult to process.

From a procedural point of view, text mining processes typically follow the CRISP-DM reference process model for data mining [8], which proposes six phases when working on a DM project: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Text mining can be distinguished from general data mining by special procedures applied in the data preparation phase, where unstructured or poorly structured text needs to be converted into organized data, structured as a table of instances (rows) described by attributes (columns). In the modeling phase, such a table of instances can be used by the standard or slightly adapted data mining algorithms to uncover interesting information hidden in the data. Two typical approaches are using clustering algorithms to find groups of similar instances, and classification rule learning algorithms to categorize new instances.

Text mining is a mature area of research offering numerous open source algorithms and NLP software libraries:

- *LATINO (Link Analysis and Text Mining Toolbox)* [9] is an open-source example of such library, which is implemented mostly under the LGPL license and is publicly available[1]. LATINO is a light-weight framework for building text mining applications, which consists of the core software library, several third party open source libraries, a collection of language resources and a range of models for tokenization, lemmatization and language detection.

- *Stanford's Core NLP Suite* [26] is a GPL-licensed framework of tools for processing English, Chinese, and Spanish. It includes several tools for tokenization (splitting of text into words), Part-of-Speech tagging, grammar parsing (identifying things like noun and verb phrases), named entity recognition, and more.

- *Natural Language Toolkit (NLTK)*[2] [10], is a Python based NLP library, which includes, similarly to the Stanford library, several capabilities for tokenizing, parsing, and identifying named entities as well as many more features.

- *TextBlob*[3] is also a Python based library for processing textual data. It is partly based on NLTK, has a simple API and addresses several tasks such as Part-of-Speech tagging, noun phrase extraction, sentiment analysis, classification, translation, etc.

- *scikit-learn*[4] [11] is a more general machine learning Python library, which features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN. It is designed to interoperate with the Python numerical and scientific libraries NumPy[5] and SciPy[6]. Moreover, it contains several approaches for extracting features from text, as well as document categorization and clustering.

- *Apache Spark*[7] is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. Similarly to scikit-learn it is also a general purpose data mining library but it supports a rich set of higher-level tools for text processing and text streaming.

- *Apache Lucene*[8] and Solr[9] are not technically targeted at solving NLP problems, but they contain a powerful number of tools for working with text ranging from advanced

[1]`https://github.com/LatinoLib/LATINO`
[2]`http://www.nltk.org`
[3]`https://github.com/sloria/textblob`
[4]`http://scikit-learn.org`
[5]`http://www.numpy.org`
[6]`http://www.scipy.org`
[7]`http://spark.apache.org`
[8]`http://lucene.apache.org`
[9]`http://lucene.apache.org/solr/`

string manipulation utilities to powerful and flexible tokenization libraries to blazing fast libraries for working with finite state automatons.

- *Apache OpenNLP*[10] uses a different underlying approach than Stanford's library: the OpenNLP project is an Apache-licensed suite of tools to do tasks like tokenization, Part-of-Speech tagging, parsing, and named entity recognition. While not necessarily state of the art any more in its approach, it remains a solid choice that is easy to get up and running.

While text mining is a mature area of research, with numerous open source algorithms and NLP software libraries available, text mining and NLP experiments are still difficult to reproduce, including the difficulty of systematic comparison of algorithms. To this end, a number of attempts have been made to develop easy-to-use workflow management systems, allowing users to compose complex processing pipelines in a visual programming manner. Section 2.2 presents some of these attempts.

2.2 Related Workflow-Management Environments

In this section we describe several text mining and natural language processing platforms, while an extensive survey of workflow management platforms, including general data mining platforms (such as RapidMiner [27], KNIME [28], Weka [29] and Orange [30]), as well as dedicated bioinformatic platforms (such as Tavaxy [31], Galaxy [32] and UGENE [33]) is out of the scope of this thesis.

Workflow management systems have become a very hot topic in the last years, mostly in the field of bioinformatics and other natural sciences. Lately, however, this trend has also spread to NLP, as evidenced by recent workshops at the main NLP conferences, e.g., the "Workshop on Language Technology Service Platforms: Synergies, Standards, Sharing" at the 9th Language Resources and Evaluation Conference (LREC 2014)[11], the "Workshop on Open Infrastructures and Analysis Frameworks for HLT"[12] at the 25th Conference on Computational Linguistics (COLING 2014) and the "Workshop on Replicability and Reproducibility in Natural Language Processing: adaptive methods, resources and software"[13] at the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015).

The current situation with workflow management systems for NLP is very fluid; some well-established systems are slowly starting to be used for NLP applications, while at the same time, new ones are being developed, specifically targeted to NLP. In this section we overview some of the more important NLP-related workflow management systems, where each platform/project is introduced with its most salient characteristics presented.

2.2.1 Taverna

The set of tools developed by the myGrid[14] team in the U.K. and used primarily for bioinformatics and other life sciences research (having in mind experiment replication) is currently probably the most advanced, richest and easiest to use workflow management (eco)system, and consists of the following main components:

- SoapLab[15] [34] which provides a convenient way to generate web services for command-line software;

[10]`https://opennlp.apache.org`
[11]`http://lrec2014.lrec-conf.org/`
[12]`http://glicom.upf.edu/OIAF4HLT/`
[13]`https://sites.google.com/site/adaptivenlp2015`
[14]`http://www.mygrid.org.uk/`
[15]`http://soaplab.sourceforge.net/soaplab2/`

- Taverna[16] [35] with its Workflow editor and Server;
- BioCatalogue[17] [36], a registry (for life sciences) where web services can be shared, searched for, annotated with tags, etc.
- myExperiment[18] [37], a social network for sharing, reusing and repurposing public workflows.

As the most important part of the myGrid offerings, we here discuss Taverna, which is conceived as a suite of tools used to design and execute scientific workflows. It combines distributed web services and/or local tools into complex analysis pipelines, which can be executed on local desktop machines or through larger infrastructures, such as supercomputers, Grids or cloud environments, using the Taverna Server. The Server allows for workflow execution from web browsers, or through third-party clients; it supports WSDL[19], REST [38], GRID[20] and cloud services, local and distributed command-line scripts, as well as other types of services, such as R-Scripts[21].

Taverna is connected to myExperiment and BioCatalogue, as well as to other service registries, such as BioVeL[22], the Biodiversity Virtual e-Laboratory. In addition, Taverna offers an Interaction Service, which enables scientists to select parameters and data during workflow execution, and the Provenance suite, which records service invocations, intermediate and final workflow results.

Taverna workflows can be designed and executed in several ways, to serve different types of workflow users. First, the Taverna Workbench—once downloaded to a local machine—provides an environment for scientists to develop new workflows and test new analysis methods, by either developing workflows from scratch, or by composing them from existing workflows. Second, workflows can be executed directly through a Taverna Server, which is an environment for serving finished workflows to a larger community of scientists. Here, a single installation of the Server provides access to a collection of workflows, normally through a web interface, called the Taverna Player; in this case, no installation or in-depth knowledge of the workflows is required, but workflows cannot be changed nor can new workflows be added to the collections.

The Taverna Workbench, as the most sophisticated means of composing workflows, needs to be first downloaded and installed on a local machine (Windows, Linux or Mac OS X). For third-party services that require a login, Taverna allows credentials to be added at run-time, or to be stored in a purpose-built credential manager. The Workbench allows users to identify and combine services by dragging and dropping them onto the workflow design panel. The services can be from third parties (using e.g., WSDL or REST), but contain a few local scripts for formatting data and managing service compatibility, known as shim services. Most workflows will need shim services as the analysis services are not usually designed to work together and, therefore, often have incompatible input and output formats. A workflow can also contain nested workflows, so workflows can be components of other workflows. Nested workflows can, for example, control retrieval of data from asynchronous services. Here the nested workflow is executed repeatedly until results are available and the control links between its output and downstream services pause the remainder of the workflow until all preceding results are available. As the workflow runs,

[16] `http://www.taverna.org.uk/`
[17] `https://www.biocatalogue.org/`
[18] `http://www.myexperiment.org/`
[19] `http://wsdl2code.googlecode.com/svn/trunk/03-Literature/WSDL/wsdl.1.1..pdf`
[20] `https://www.ogf.org/documents/GFD.72.pdf`
[21] `https://www.r-project.org/`
[22] `http://www.biovel.eu/`

the results panel shows progress through the workflow and iterations over data, as well as any errors if there are problems with executions.

The Taverna Server, which executes workflows, can also be downloaded and configured to run with or without login restrictions. It is written in Java, has to be installed on Unix and uses Tomcat with, for secure mode, HTTPS and SSL host certificate. There are various public installations of the Taverna Server, with the best known being the already mentioned BioVeL portal with workflows from the area of biodiversity.

As mentioned, Taverna is currently the most developed and advanced (open source) workflow management system, with a host of features and connection capabilities, including fine-grained access management. It is also very popular, e.g., myExperiment currently contains over 2,000 Taverna workflows. By far the largest part of the user community is from the field of bioinformatics and other life sciences, where Taverna workflows are typically used in the areas of high-throughput analyses or for evidence gathering methods involving data mining.

Given the power and versatility of Taverna and other myGrid platforms, it is surprising that—apart from a few basic workflows submitted by various individuals—there are few public NLP workflows have been so far implemented in it. In connection with biomedical informatics, there is one published experiment in text mining [39], which has given rise to further research and there is also one platform that makes use of the myGrid building blocks for the purposes of NLP, which is the subject of Section 2.2.2.

2.2.2 PANACEA

In PANACEA[23] [40], a FP7 project that ran 2010–2012, the objective was to automate the stages involved in the acquisition, production, updating and maintenance of language resources required by machine translation systems, and by other applications for processing of natural language.

The architecture of this factory is based on deploying NLP tools as web services using SoapLab to generate them for command-line software, this being the standard mode of invocation for most current NLP tools. These individual services can then be combined in the Taverna Workbench and deployed on a Taverna Server.

In the scope of PANACEA various enhancements have been made to the underlying technology, e.g., the possibility to limit in SoapLab the amount of direct data that SOAP messages can transfer; various bugs were also identified and reported to the developers. A common interchange format for the language resources (esp. annotated corpora) was also defined [41], in the first instance XCES [42], because that was previously used by most of the project partners, but in the final version the format moved to the more current Linguistic Annotation Format (LAF) and Graph-based Format for Linguistic Annotations (GrAF) developed in the scope of the ISO/TC37/SC4 technical committee [43]. Finally, the IPR and other legal issues connected to sharing possibly proprietary tools and esp. resources were also considered in PANACEA [44].

The concrete results of the project are made available via ELDA[24], the European Evaluations and Language resources Distribution Agency, and consist of:

- the PANACEA Registry[25], currently describing 163 services (not all freely available)
- the PANACEA MyExperiment[26] installation, which allows exploring workflows, but allows executing them only after registration.

[23] `http://panacea-lr.eu/`
[24] `http://www.elda.org/`
[25] `http://registry.elda.org/`
[26] `http://myexperiment.elda.org/`

The actual web services mostly run on machines (SoapLab or Taverna servers) of the project partners. It should be noted that the ELDA installation is made with an eye to commercializing the platform.

2.2.3 Argo

Tsujii Lab at the University of Tokyo had a long tradition in combining NLP with biomedical informatics. For example, they were the creators of the GENIA corpus [45], the first and best known biomedical corpus annotated with linguistic information. In connection with their work on NLP for bioinformatics a workflow for text mining U-compare[27] [46], which includes NLP annotation services was developed. U-compare is implemented as an independent Java application, using the Apache UIMA[28] framework. This work was later intergrated into Taverna, in the already mentioned work by [39].

Recently, a new workflow management system has been developed on the basis of U-compare, now in the context of the National Centre for Text Mining (NaCTeM) at the University of Manchester, called Argo[29] **Argomain**, [47]. Argo offers the usual array of features, accessed through a browser based interface: the user can upload documents, compose workflows and execute them. A novelty introduced by Argo is that workflows can have interactive components as well, where the execution of the workflow pauses to receive input from the user. This is useful for workflows which allow for manual annotation of corpora by the user, and Argo offers several such workflows. However, Argo does not seem to have any sophisticated utilities for cataloguing available web services or workflows, nor a system of access permissions.

As far as its architecture goes, Argo continues to be based on UIMA, and uses REST for communication between the components, so other services or workflows can call Argo defined web services as well. It supports import and export (deserializations and serializations) into RDF[48], which is the de-facto language of the Semantic Web. The requirement that web services need compatible I/O formats is in Argo resolved with a powerful system based on cascaded finite-state transducers called Type Mapper [49], which allows for developing needed shim services.

2.2.4 WebLicht

The European Research Infrastructure CLARIN[30] aims to provide the researchers unified single sign-on access to a platform which integrates language-based resources and advanced tools. This is to be implemented by the construction and operation of a shared distributed infrastructure that aims at making language resources, technology and expertise available to the humanities and social sciences research communities. CLARIN is a distributed data infrastructure, with national sites in a number of European countries. These sites provide access to language data repositories, to services and workflows to work with language data, and to expertise that enables researchers to use these resources and tools.

In the scope of national CLARIN portals, various workflows have been developed, with the German WebLicht[31] [50] being the most advanced. While WebLicht is, in some respects, similar to PANACEA, the focus here is not on providing web services for human language technology research, but rather producing annotated corpora for use in linguistic

[27]`http://u-compare.org/`
[28]`https://uima.apache.org/`
[29]`http://argo.nactem.ac.uk/`
[30]`http://www.clarin.eu/`
[31]`http://weblicht.sfs.uni-tuebingen.de/`

research. Nevertheless, the methods are similar, as on both platforms most web services are devoted to the linguistic annotation of textual input.

WebLicht does not, as does PANACEA, use the myGrid tools and platforms but rather developed its own infrastructure, starting from the wrappers to make the included tools into RESTful web services, to its centralized repository, and the web editor enabling the composition of the workflows (or toolchains, as they are known in WebLicht, as they are typically limited to one input and one output). As opposed to PANACEA, WebLicht does cover all the standard linguistic annotation steps, for a number of languages, and with often several tools being available for an annotation step. The WebLicht repository includes information about the type of inputs and outputs of individual services, which allows controlling the workflow construction process in the editor to allow connecting only tools that have matching I/O specifications. For example, a Part-of-Speech tagger needs a tokenizer to be applied to a text before it can be called. As in PANACEA, WebLicht also imposes standards on the I/O interchange format: it uses TCF (Text Corpus Format) [51], a format similar but slimmer than ISO LAF/GrAF, but also provides conversion to LAF/GrAF. With TCF the text and annotations are in one XML file, but with stand-off annotation, with each processing step adding a layer of annotation.

The web-based WebLicht editor allows the construction of workflows and their invocation (after a CLARIN(-DE) recognized log-in) and viewing or saving the results. WebLicht has a dedicated viewer, which allows displaying an annotated corpus in tabular format (for tokens and their annotations), lists (for sentences) or as a graphic (for parse trees).

While the WebLicht platform is not open source, the initiative is open to adding new partners who are interested in contributing tools and services. This typically involves wrapping the tools to be contributed to make them RESTful and to take care of I/O requirements, installing this web service on a local machine and then registering them to the WebLicht central repository. Currently, such third-party web services are provided for the Finnish language, by the University of Helsinki, and for the Romanian language, by their Academy of Sciences.

2.2.5 Language Grid

The Language Grid[32] [52] is a multilingual Internet service platform, developed by Language Grid Project (started in 2006) at the Japanese National Institute of Information and Communications Technology. The Language Grid is based on the service-oriented architecture (SOA), a web-oriented version of the pipeline architecture typically employed by NLP tools. As other workflow management systems it provides three main functions: language service registration and deployment, language service search, and language service composition and execution. Importantly, Language Grid also offers access to a large number of language resources such as online dictionaries and bi-lingual corpora.

In contrast to e.g., myGrid, geared towards running and reproducing scientific experiments, the Language Grid is much more application oriented, with a focus on enabling communication in multilingual communities (via machine translation), the best known example being the support of farming in rural communities in Vietnam, by enabling computer mediated communication between Vietnamese youth and Japanese experts in agriculture. This is also reflected in its architecture, where the users, services and workflows (or "composite services") are centrally administered. And while running such web services is easy with the provided graphical interface, constructing them is more complicated: workflows are composed using WS-BPEL (Web Service Business Process Execution Language) as XML files, rather than in a dedicated web based or locally installed visual editor. Al-

[32]http://langrid.org/

though Eclipse does provide a visual BPEL editor, which can be used for this process, workflow construction is still more complicated than with e.g., Taverna.

The Language Grid is Open source and the first European node, known as Linguagrid[33] [53] was established in Italy in 2010.

2.2.6 LAPPS Grid

The Language Grid Server also serves as the basis for the U.S. Language Application (LAPPS) Grid project[34] [54]. LAPPS is especially interesting in three aspects. First, it uses advanced standards for data encoding and exchange, in particular the JSON-based serialization for Linked Data (JSON-LD). The JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format that defines a small set of formatting rules for the portable representation of structured data. Because it is based on the W3C Resource Definition Framework (RDF), JSON-LD is simple to map to and from other graph-based formats, in particular the already mentioned ISO LAF/GrAF [43]. It also enables services to reference categories and definitions in web-based repositories and ontologies, such as those of ISOcat[35]. Second, it uses the Open Advancement approach (developed in the making of IBM's Watson [55]) for component- and application-based evaluation, that has been successful in enabling rapid identification of frequent error categories within modules and documents, thus contributing to more effective investment of resources in both research and application assembly. The LAPPS Grid thus envisions scenarios where it is possible for users to rapidly (re)configure and automatically evaluate a (changed) pipeline on a chosen dataset and metrics. Third, workflows are planned to incorporate a comprehensive model for addressing constraints on the intellectual property used in the LAPPS Grid, making it maximally open to users ranging from open source developers to commercial users of language services.

In this section we described several text mining and natural language processing platforms. In summary, none of the described text analytics workflow management systems offers combination of components from different contexts (e.g., using clustering in relational data mining) and from different software sources (e.g., building ensemble classifiers from different libraries), while still maintaining a high level of simplicity, providing full-time accessibility, enabling methodology and result sharing and reuse, and being open source. The lack of such a platform motivated us to develop the TextFlows platform, presented in this thesis, which is oriented towards text analytics and offers a number of algorithms for text mining and natural language processing. The platform is implemented as a cloud-based web application and aims to overcome various deficiencies of similar text analytics platforms, providing novel features that should be beneficial to the text mining community.

2.3 Literature-Based Cross-Domain Knowledge Discovery

Among varieties of information that can be extracted from text, part of this thesis focuses on a specific type of information extraction task referred to as bisociation discovery, and in particular bisociative literature mining. This part of research has scientific grounds in the literature on domain-crossing associations, called bisociations, introduced in Koestler's book "The Act of Creation" [13]. According to Koestler, bisociative thinking occurs when a problem, idea, event or situation is perceived simultaneously in two or more "matrices of

[33]`http://www.linguagrid.org/`
[34]`http://lapps.anc.org/`
[35]`http://www.isocat.org/`

though" or domains. When two matrices of thought interact with each other, the result is either their fusion in a novel intellectual synthesis or their confrontation in a new aesthetic experience. He regarded many different mental phenomena that are based on comparison (such as analogies, metaphors, jokes, identification, anthropomorphism, and so on) as special cases of bisociation. More recently, this work was followed by the researchers interested in the so-called bisociative knowledge discovery [14], where—according to Berthold—two concepts are bisociated if there is no direct, obvious evidence linking them and if one has to cross different domains to find the link, where a new link must provide some novel insight into the problem addressed.

In the area of bisociative literature mining, also called *literature-based discovery (LBD)*, Smalheiser and Swanson [56], [57] presented a new approach to discover unknown relations between previously unrelated concepts. As suggested in their work, this can be done by identifying interesting bridging terms (B-terms) appearing in two disparate literatures (sets of documents) and bearing a potential of indirectly connecting the two concepts (described through the two document sets) under investigation. Smalheiser and Swanson developed an online system ARROWSMITH, which takes as input two sets of documents from disjoint domains A and C and lists the terms that are common to A and C; the resulting bridging terms are further investigated for their potential to generate new scientific hypotheses. The major focus in literature-based discovery has been on the closed discovery process, where both A and C have to be specified in advance [58].

Inspired by this early work, literature-based discovery approaches were further developed and successfully applied to different problems, such as finding associations between genes and diseases [59], diseases and chemicals [60], and others. Holzinger et al. [61] describe several quality-oriented web-based tools for the analysis of biomedical literature, which include the analysis of terms (biomedical entities such as disease, drugs, genes, proteins and organs) and provide concepts associated with a given term. A recent approach by Kastrin et al. [62] is complementary to the other LBD approaches, as it uses different similarity measures (such as common neighbors, Jaccard index, and preferential attachment) for link prediction of implicit relationships in the Semantic MEDLINE network.

Early Swanson's work has shown that databases such as PubMed can serve as a rich source of yet hidden relations between usually unrelated topics, potentially leading to novel insights and discoveries. By studying two separate literatures—the literature on migraine headache and the articles on magnesium—Swanson [63] discovered "Eleven neglected connections", all of them supportive for the hypothesis that magnesium deficiency might cause migraine headache. Swanson's literature mining results have been later confirmed by laboratory and clinical investigations. This well-known example has become a gold standard in the literature mining field and has been used as a benchmark in several studies, including those presented in [17], [19], [58], [64]–[67]. The experimental data used to test the cross-domain knowledge discovery methodology proposed in this work are papers from the combined migraine-magnesium domain, studied extensively by Swanson and his followers, as well as the combined autism-calcineurin domain pair explored in [17], [19], [66], [68].

The main challenge in literature-based discovery research is how to narrow the list of candidate bridging terms, which act as links between the domains of interest. In the RaJoLink methodology [69] the list of interesting terms is effectively filtered according to MeSH (Medical Subject Headings) categories; in the next step the expert checks which terms (from the usually long list of resulting terms) seem promising. The developers of the CrossBee system [17]–[19] explore a specific form of bisociation by ranking terms that appear in documents, which represent bisociative links between concepts of different domains. Term ranking is based on the voting of an ensemble of heuristics. The resulting ranked list of potential domain bridging terms enables the user to start inspecting these

B-terms with top-ranked terms, which should result in higher probability of finding observations that may lead to the discovery of new links between different domains. Research in literature mining, conducted by Petrič et al. [66] and Sluban et al. [67], suggests that bridging terms are more frequent in documents that are in some sense different from the majority of documents in a given domain. For example, in [67] it was shown that such documents, considered being outlier documents of their own domain, contain a substantially larger amount of bridging-linking terms than the normal, non-outlier documents.

In part of the thesis we continue the development of cross-domain bisociation exploration engine CrossBee [17]–[19]. CrossBee is an off-the-shelf solution for finding bisociative terms bridging two domains. The CrossBee web application is built on top of the CrossBee library created in previous work of Juršič et al. [17], [70]. The CrossBee human-computer interaction (HCI) functionality includes the following facilities: (a) *Performance evaluation* that can be used to measure the quality of results, e.g., through plotting ROC curves when the actual bridging terms are known in advance. (b) *Marking of high-ranked terms* by emphasizing them, thus making them easier to spot throughout the application. (c) *B-term emphasis* can be used to mark the terms predefined as B-terms by the user. (d) *Domain separation* colors all the documents from the same domain with the same color, making an obvious distinction between the documents from the two domains. (e) *User interface customization* enables the user to decrease or increase the intensity of the following features: high-ranked term emphasis, B-term emphasis and domain separation; this facility was introduced to enable the user to set the intensity of these features, given that in cooperation with the experts we discovered that some of them like the emphasizing features while others do not.

In summary, infrastructures and workflows that could effectively support bisociative knowledge discovery are still immature and this thesis aims at bridging this scientific gap through advanced text analytics workflow development including the use of shared vocabularies in text processing.

2.4 Relational Data Mining and Propositionalization

Inductive Logic Programming (ILP) and Relational Data Mining (RDM) algorithms are characterized by the ability to use background knowledge in learning relational models or patterns [20]–[23], as by taking into account additional relations among the data objects the performance of data mining algorithms can be significantly improved.

Propositionalization [24], [25] is an approach to ILP and RDM, which offers a way to transform a relational database into a propositional single-table format. In contrast to methods that directly induce relational patterns or models, such as Aleph [71] and Progol [72], propositionalization algorithms transform a relational problem into a form which can be solved by standard machine learning or data mining algorithms. Consequently, learning with propositionalization techniques is divided into two self-contained phases: (1) relational data transformation into a single-table data format and (2) selecting and applying a propositional learner on the transformed data table. As an advantage, propositionalization is not limited to specific data mining tasks such as classification, which is usually the case with ILP and RDM methods that directly induce models from relational data.

The transformation to a single-table format can be achieved for the so-called individual-centered relational databases [73], i.e. databases that have a clear notion of an individual. For example, the East-West trains challenge [74], where the task is to classify the trains as East-bound or West-bound, is a well-known domain in which individuals are clearly identified: each train is a single individual related with one or more cars that have different characteristics.

Most of the related work involves propositionalization through first-order feature construction [24], [25], [75], [76], where the algorithms construct complex first-order features, which then act as binary attributes in the new propositional representation of examples. One of the first propositionalization algorithms, LINUS [24], generates features that do not allow recursion and newly introduced variables. An improvement of LINUS, named SINUS [77], incorporates more advanced feature construction techniques inspired by feature construction implemented in 1BC [73]. RSD [75] is a relational subgroup discovery algorithm composed of two main steps: the propositionalization step and the subgroup discovery step, where the output of the propositionalization step can be used as input to other propositional learners. RSD effectively produces an exhaustive list of first-order features that comply with the user-defined mode constraints, similar to those of Progol [72] and Aleph [71]. Furthermore, RSD features satisfy the connectivity requirement, which imposes that no constructed feature can be decomposed into a conjunction of two or more features.

RELAGGS [78], which stands for *rel*ational *agg*regation, is a propositionalization approach, which uses the input relational database schema as a basis for a declarative bias and aims to use optimization techniques usually used in relational databases (e.g., indexes). Furthermore, the approach employs aggregation functions in order to summarize non-target relations with respect to the individuals in the target table.

An early experimental comparison of propositionalization techniques was reported in [79], where RSD, SINUS and RELAGGS algorithms were compared.

Other means of propositionalization include stochastic propositionalization [25], which employs a search strategy similar to random mutation hill-climbing: the algorithm iterates over generations of individuals, which are added and removed with a probability proportional to the fitness of individuals, where the fitness function used is based on the Minimum Description Length (MDL) principle.

Safarii[36] is a commercial multi-relation data mining tool. Safarii, which is extensively described in [80], offers a unique pattern language that merges ILP-style structural descriptions as well as aggregations. Furthermore, Safarii comes with a tool called ProSafarii, which offers several preprocessing utilities, including propositionalization via aggregation.

Ceci et al. [81] investigate spatial classification which uses a propositionalization approach which constructs features using spatial association rules to produce an attribute-value representation. They compare the approach to a structural approach using an extended Naive Bayes classifier. They report an advantage of the structural alternative in terms of accuracy, while the propositional approach performs faster. Ceci et al. [82] present two emerging patterns based classifiers that work in the multi-relational setting: one uses a heuristic evaluation function to classify objects, while the other is based on a probabilistic evaluation. The main result of the study is that both approaches perform better than associative classification to which they were compared.

Kuželka and Železný developed RelF [76], which constructs a set of tree-like relational features by combining smaller conjunctive blocks. The novelty is that RelF preserves the monotonicity of feature reducibility and redundancy (instead of the typical monotonicity of frequency), which allows the algorithm to scale far better than other state-of-the-art propositionalization algorithms.

An approach that is related to propositionalization is presented in [83]. The authors propose a strategy of multi-relational learning where they neither upgrade a propositional learner to work with multiple relations or propositionalize the relations. Instead, their approach learns from multiple views (feature sets) of a RDB and then integrates the in-

[36]`http://www.kiminkii.com/safarii.html`

dividual view learners to construct a final model. Their approach exhibits comparable classification accuracies compared to related approaches, and a faster runtime.

Recently, a propositionalization technique called Bottom Clause Propositionalization (BCP) was introduced in [84]. It was integrated with C-IL^2P [85]; the combined system, named CILP++, achieves accuracy comparable to Aleph, while being faster. Compared to RSD, BCP is better in terms of accuracy when using a neural network and similar when using C4.5.

None of the described approaches studies the means of combining relational data mining and text mining, which is the approach proposed in this thesis. Moreover, infrastructures and workflows that would effectively support such transformations do not exist and this thesis aims at bridging this scientific gap.

Chapter 3

TextFlows: A Cloud-Based Text Mining Platform

This chapter presents TextFlows, a web-based text mining and natural language processing platform supporting workflow construction, sharing and execution. The platform enables visual construction of text mining workflows through a web browser, and the execution of the constructed workflows on a processing cloud. This makes TextFlows an adaptable infrastructure for the construction and sharing of text processing workflows, which can be reused in various applications.

The chapter is structured as follows. The motivation for developing a new web-based text mining platform TextFlows is presented in Section 3.1. In Section 3.2 we present the technical background and architecture of the TextFlows platform, along with its key text mining components. The architecture of the system is presented in detail along with specific data structures that allow efficient text mining in a workflow environment. The concept of workflows, their implementation, execution and sharing are presented in Section 3.3, while Section 3.4 describes the widget repository and the implemented modules of the platform. In Section 3.5 we compare TextFlows with other state-of-the-art workflow management systems along several dimensions that affect the usefulness of each system, while Section 3.6 explains the relationship between TextFlows and ClowdFlows.

3.1 Motivation for a Cloud-Based Text Mining Platform

The TextFlows platform described in this chapter is a new open source, web-based text mining platform that supports the design and composition of scientific procedures implemented as executable workflows. As a fork of ClowdFlows [12], TextFlows has inherited its service-oriented architecture that allows the user to utilize arbitrary web services as workflow components. TextFlows is oriented towards text analytics and offers a number of algorithms for text mining and natural language processing. The platform is implemented as a cloud-based web application and attempts to overcome various deficiencies of similar text analytics platforms, providing novel features that should be beneficial to the text mining community. In contrast to existing text analytics workflow management systems, the developed platform is the only one with all the following properties. It is simple (i.e. enables visual programming, is web-based and requires no installation), enables workflow sharing and reuse, and is open source. Moreover, the platform enables combining workflow components (called “widgets”) from different contexts (e.g., using clustering in relational data mining) and from different software sources (e.g., building ensemble classifiers from different libraries). To do so, it provides a unified input-output representation, which enables interoperability between widget libraries through automated data type conversion. It uses a common text representation structure and advocates the usage of 'hubs' for algorithm execution.

The TextFlows platform is publicly available at `http://textflows.org`, while its source code is available at `https://github.com/xflows/textflows` under the MIT License. Detailed installation instructions are provided with the source code. After setting up a local TextFlows instance, advanced users can also implement and test their own algorithms. Improvements to the code can also be pushed to the main Git code repository via pull requests. The committed changes are reviewed by the TextFlows core team and merged into the master branch.

TextFlows is a web application which can be accessed and controlled from anywhere while the processing is performed in a cloud of computing nodes. TextFlows differs from most comparable text mining platforms in that it resides on a server (or cluster of machines) while its graphical user interface for workflow construction is served as a web application through a web browser. The distinguishing feature is the ease of sharing and publicizing the constructed workflows, together with an ever growing roster of reusable workflow components and entire workflows. As not only widgets and workflows, but also data and results can be made public by the author, TextFlows can serve as an easy-to-access integration platform both for various text mining workflows but also for experiment replicability. Each public workflow is assigned a unique URL that can be accessed by anyone to either repeat the experiment, or to use the workflow as a template to design another, similar, workflow.

Workflow components (widgets) in TextFlows are organized into packages, which allows for easier distributed development. The TextFlows packages implement several text mining algorithms from LATINO[1] [9], NLTK[2] [10] and scikit-learn[3] [11] libraries. Moreover, TextFlows is easily extensible by adding new packages and workflow components. Workflow components of several types allow graphical user interaction during run-time and visualization of results by implementing views in JavaScript, HTML or any other format that can be rendered in a web browser (e.g., Flash, Java Applet).

3.2 Platform Architecture and Components

This section presents the TextFlows platform, together with its architecture and main components of the system, as well as introduces the graphical user interface. Like its predecessor data mining platform ClowdFlows [12], TextFlows can also be accessed and controlled from a web browser, while the main processing is performed on a cloud of computing nodes.

3.2.1 Platform Architecture

In software engineering, terms *front-end* and *back-end* are used to distinguish the separation between a presentation layer (the client side) and a data access layer (the server side), respectively. Figure 3.1 shows the TextFlows architecture, which is logically separated into the front-end (in pink) and the back-end (in blue). The back-end comprises a relational database for storing workflows, workers for task execution and a broker for delegating tasks from different queues to workers which can reside on different clusters of machines. The front-end is designed for user interaction with workflows through the graphical user interface in a web browser.

[1]LATINO (Link Analysis and Text Mining Toolbox) is open-source—mostly under the LGPL license—and is available at `https://github.com/LatinoLib/LATINO`

[2]`http://www.nltk.org`

[3]`http://scikit-learn.org`

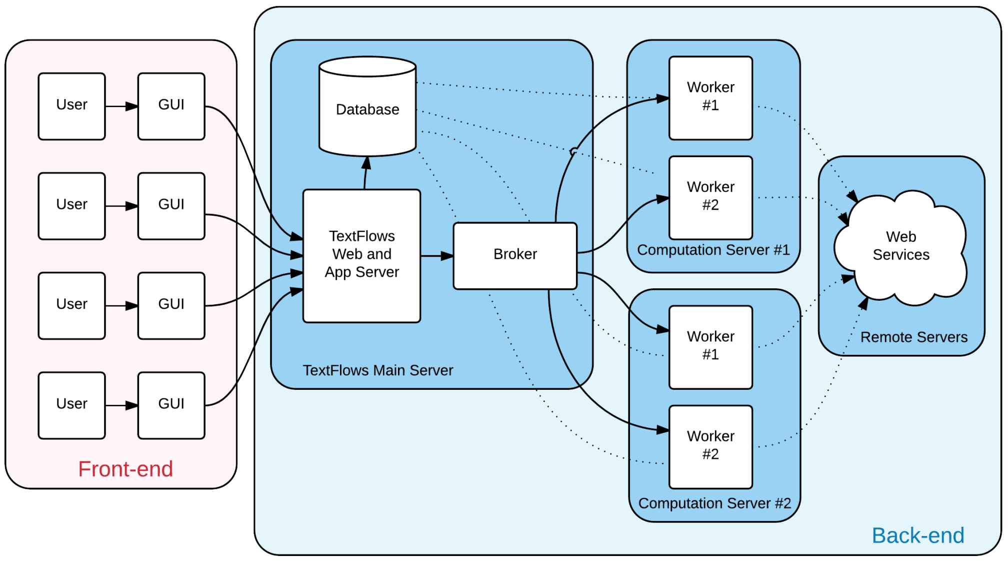


Figure 3.1: An overview of the TextFlows architecture, separated into the front-end (in pink) and the back-end (in blue).

The back-end of the TextFlows platform uses Django[4], a Python-based open source high-level web framework. Django follows the model–view–controller architectural design, which encourages rapid development, and includes an object-relational mapper and a template engine for efficient code generation. The object-relational mapper provides an interface which links objects to a database. This provides support for several database systems, such as PostgreSQL, MySQL, SQLite and Oracle. PostgreSQL is used in the public installation of TextFlows.

Without additional extensions Django is synchronous, sometimes also described as blocking. This means that a HTTP request will not be returned until all processing is complete. Though this is the expected behavior usually required in web applications, in the case of TextFlows we need tasks to run in the background without blocking. Furthermore, different system environment requirements of implemented libraries dictate that the TextFlows platform must be distributed across multiple machines (e.g., the LATINO library uses the .Net framework and performs best on Windows operating systems). TextFlows task queues are used as a mechanism to distribute work across threads and machines. This is performed via Celery[5], which is a task queue system based on distributed message passing. Celery is focused on real-time operation, but supports scheduling as well. Dedicated worker processes monitor task queues for new work to perform and active workers execute different tasks concurrently on multiple servers. Tasks can be executed asynchronously (in the background) or synchronously (wait until ready). A Celery system can consist of multiple workers and brokers, thus supporting high availability and horizontal scaling.

Celery communicates via messages and uses a message broker to mediate between clients and workers. To initiate a task, a client adds a message to the queue, which the broker

[4]`https://www.djangoproject.com`
[5]`http://www.celeryproject.org`

then delivers to a worker. The system used as a broker in TextFlows is RabbitMQ[6], a complete and highly reliable enterprise messaging system based on the Advanced Message Queuing Protocol (AMQP). It offers not only exceptionally high reliability, but also high availability and scalability, which is vital for the TextFlows platform.

TextFlows uses the PySimpleSoap library[7] for integrations of web services as workflow components. PySimpleSoap is a lightweight Python library which provides an interface for client and server web service communication. Using PySimpleSoap we cannot only import WSDL web services as workflow components, but also expose entire workflows as WSDL web services.

The client side of the TextFlows platform consists of operations that involve user interaction primarily through the graphical user interface (GUI) in a modern web browser. The graphical user interface is implemented in HTML and JavaScript, with an extensive use of the jQuery library[8]. The jQuery library was designed to simplify client-side scripting and is the most popular JavaScript library in use today[9]. On top of jQuery we use the interaction library jQuery-UI[10], which is a collection of GUI modules, animated visual effects, and themes. This library supports the option to make elements in the graphical user interface draggable, droppable, and selectable, which are the features supported by the TextFlows workflow canvas (cf. Section 3.3).

3.2.2 Key Text Mining Concepts in TextFlows

The key concepts used in text mining and natural language processing are a document collection (or corpus), a single document (or text), and document features (or annotations) [1]. The following sections describe the model of corpora, documents and annotations in TextFlows. When designing TextFlows, emphasis was given to common representations that are passed among the majority of widgets: each TextFlows document collection is represented by an instance of the *AnnotatedDocumentCorpus (ADC)* class, a single document is an instance of the *AnnotatedDocument* class, while the features are instances of the *Annotation* class.

Annotated Corpus

A document collection is any grouping of text documents that can be used in text analytics. Even though the size of a collection may vary from a few to millions of documents, from the text analysis perspective, more is better. In TextFlows, the Python class that represents a corpus of documents is called *AnnotatedDocumentCorpus (ADC)*. Every ADC instance contains not only a collection of documents which are part of this corpus but also the features that provide additional information about the corpus (e.g., authors, date of collection, facts and notes about the dataset, etc.). The features are stored in a simple key-value Python dictionary, where keys are strings and the values can store any Python object.

Annotated Document

In TextFlows a single textual data unit within a collection—a document—is represented by the class *AnnotatedDocument*. An *AnnotatedDocument* object contains the text of the entire document, which may vary in size, e.g., from a single sentence to a whole book.

[6] `http://www.rabbitmq.com`
[7] `https://code.google.com/p/pysimplesoap/`
[8] `http://jquery.com`
[9] `http://w3techs.com/technologies/overview/javascript_library/all`
[10] `http://jqueryui.com`

Similarly to *AnnotatedDocumentCorpus*, *AnnotatedDocument* instances in TextFlows also contain features which may provide information about a single document (e.g., author, date of publication, document length, assigned keywords, etc.).

Annotation

In TextFlows *Annotation* instances are used to mark parts of the document, e.g., words, sentences or terms. Every *Annotation* instance has two pointers: one to the start and another to the end of the annotation span in the document text. These pointers are represented as the character offset from the beginning of the document. *Annotation* instances also have a type attribute, which is assigned by the user and is used for grouping annotations of similar nature. As described in detail in Section 3.4, annotations can also contain key-value dictionaries of features, which are used by various taggers to annotate parts of document with a specific tag, e.g., annotations of type "token" that have a feature named "StopWord" with value "true", represent stop words in the document.

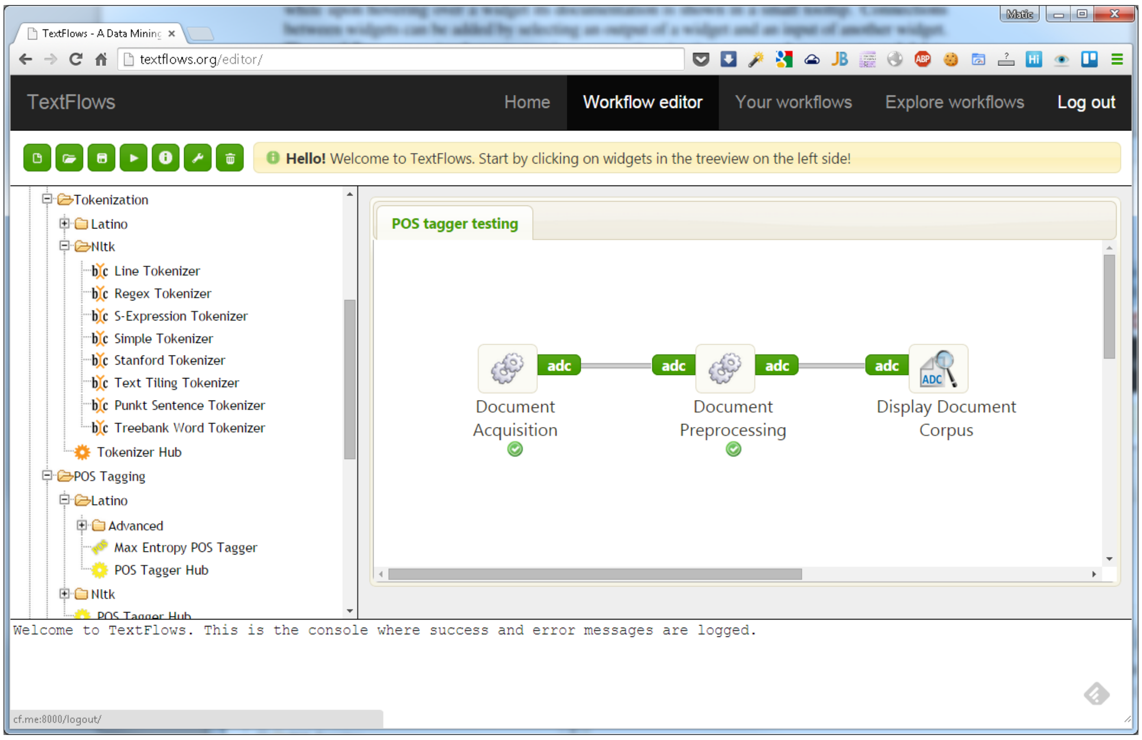
3.3 The Concept of Workflows

The workflow model in the TextFlows platform consists of an abstract representation of workflows and workflow components. A workflow is an executable graphical representation of a complex procedure. The graphical user interface (GUI) used for constructing workflows follows a visual programming paradigm which simplifies the representation of complex procedures into a spatial arrangement of building blocks. The most basic unit component in a TextFlows workflow is a processing component, which is represented as a widget in the graphical representation. Considering its inputs and parameters, every such component performs a task and outputs the results. Different processing components are linked via connections through which data are transferred from a widget's output to another widget's input. Additional inputs for a widget are its parameters, which the user enters into the widget text fields. The graphical user interface implements an easy-to-use way of arranging widgets on a canvas to form a graphical representation of a complex procedure.

The TextFlows graphical user interface, illustrated in Figure 3.2, consists of a widget repository and a workflow canvas. The widget repository is a set of widgets ordered in a hierarchy of categories. Upon clicking on a widget in the repository, the widget is added as a building block to the workflow canvas. While hovering over a widget its documentation is shown in as a tooltip. Connections between widgets can be added by clicking on an output of a widget and then on an input of another widget. The workflow canvas implements moving, connecting, and issuing commands to execute or delete widgets. Every action on the workflows canvas causes an asynchronous HTTP POST request to be sent to the server. After the requested operation is validated on the server, a success or error message with additional information is passed to the user interface. An example of such a validation is checking for cycles in the workflows when connecting widgets.

3.3.1 Workflow Execution Engine

The workflow execution engine is responsible for executing the workflow widgets in the predefined order. Two such engines are integrated into the TextFlows platform: a server-side implementation in Python and a client-side implementation in JavaScript. Sub-parts of workflows in subprocesses and loops are executed by the server-side Python implementation, while the top-level workflow is executed from the user interface (when the user actually needs to see the order of the executed widgets in real time) are processed by the client-side JavaScript implementation. The former shows results only after their complete



Welcome to TextFlows. This is the console where success and error messages are logged.

Figure 3.2: A screenshot of the TextFlows GUI opened in Google Chrome. On the top there is a toolbar for saving, deleting entire workflows. On the left is the widget repository giving a list of available widgets grouped by their functionality. By clicking on a widget in the repository it is added to the workflow construction canvas which is on the right. The console for displaying success and error message is located on the bottom of the interface.

execution, while the latter allows showing results of the execution in real time. With the Python implementation the server receives only one HTTP request for the entire part of the workflow, therefore multiprocessing had to be implemented manually. On the other hand, when the widget execution order is regulated with the JavaScript engine, it perpetually checks for executable widgets and processes them. Executable widgets are widgets, which either have no predecessors or their predecessors have already been successfully executed. Whenever two or more independent widgets can be executed at the same time they are asynchronously executed in parallel. Each widget is executed on the server through a separate asynchronous HTTP request. Every request is handled by the server separately and executes a single widget. When a widget is prepared to be executed, a task is added to a relevant task queue; as some widgets have special library requirements or even system requirement they are executed in a separated task queue with its dedicated workers. Celery communicates via messages and uses a message broker to find a suitable worker to which the task is delivered to. Dedicated worker processes monitor task queues for new work to perform. When the task is executed its result is saved into the database and returned to the client. The execution of a workflow is considered complete when there are no more executable or running widgets in the workflow.

3.3.2 Workflow Sharing

Workflows in TextFlows are processed and stored on remote servers from where they can be accessed from anywhere, requiring only an Internet connection. By default each workflow can only be accessed by its author, although (s)he may also choose to make it publicly available. The TextFlows platform generates a specific URL for each workflow that has been saved as public. Users can then simply share their workflows by publishing the URL. Whenever a public workflow is accessed by another user, a copy of the workflow is created on the fly and added to their private workflow repository. The workflow is copied with all the relevant data to ensure that the experiments can be repeated. In this way, the users are able to tailor the workflow to their needs without modifying the original workflow.

3.4 The Widget Repository

This section presents the TextFlows widget repository. First we describe different types of widgets, followed by the presentation of widgets based on their functionality as they appear in the TextFlows widget repository.

Widgets in TextFlows are separated into four groups based on their type:

- *Regular widgets* perform specific tasks that transform the data from their inputs and parameters to data on their outputs, and provide success or error messages to the system. In the back-end such widgets are represented as Python functions, which receive (on every widget execution) a Python dictionary of inputs and parameters as widget arguments, perform a specific task and return a dictionary of outputs. The function is called each time the widget is executed. Widgets that implement complex long-running procedures can also display a progress bar, which shows the progress to the user in real time.

- *Visualization widgets* are extended versions of regular widgets as they also provide the ability to render an HTML template with JavaScript to the client's browser, which is useful for data visualizations and presentation of a more detailed feedback to the user. Visualization widgets differ from regular widgets by a secondary Python function which controls the rendering of the template. This function is only invoked when the widget is executed using the JavaScript execution engine, i.e. when it is not part of a subprocess.

- *Interactive widgets* are extensions of regular widgets as they pop-up an additional window during execution through which the user can interact with or manipulate the data (an example of an interactive widget is shown in Figure 4.12). The entire procedure is implemented using three Python functions. The first function receives the widget's inputs and initialization parameters as its arguments and prepares the data for the second function. The second function renders (using an HTML template) a pop-up window that prompts the user to manipulate the data. The final function uses the user inputs, as well as the widget's inputs and parameters to produce the final output of the widget.

- *Workflow control widgets* provide additional workflow controls which allow the user to combine different workflow components into subprocesses, and provide different types of iterations through data (e.g., iteration through a list of classifiers, applying a classifier to all folds in cross-validation, etc.). This group of widgets consists of: *Subprocess*, *Input*, *Output*, *For Input*, *For Output*, *Cross Validation Input* and *Cross Validation Output* widgets. Whenever a *Subprocess* widget is added to a workflow, an initially empty workflow with no inputs and outputs is created. Then, when an *Input* or *Output* widget is attached to a subprocess workflow, an input or output is automatically added to the subprocess widget itself. This way Workflows can be indefinitely nested.

Two additional variations of the input and output widgets exist in TextFlows. When a subprocess contains the *For Input* and *For Output* widgets, the workflow execution engine will emulate a for loop by attempting to break down the object on the input and executing the subprocess workflow once on every iteration. Using these controls a subprocess can be iteratively executed on a list. Similarly, if the user opts to use the *Cross Validation Input* and *Cross Validation Output* widgets, the input data will be divided into the training and test dataset according to the selected number of folds; if the input data is labeled, stratified cross-validation [86] is performed.

The widget repository shows the hierarchy of all the widgets currently available in the TextFlows platform, grouped by their functionality. There are four top-level categories:

- *Text mining widgets:* a collection of implemented text mining widgets; these widgets are further divided based on their text mining functionality.
- *Basic widgets:* widgets that are responsible for creating and manipulating simple objects such as strings and integers.
- *Subprocess widgets:* a collection of workflow control widgets, which are required for visual programming of complex procedures.
- *WSDL imports:* workflow components representing the WSDL web services that the user has imported.

In the following sections we present in more detail the text mining widgets based on their functionality in the order of appearance in the TextFlows widget repository.

3.4.1 Corpus and Vocabulary Acquisition

Document acquisition (import) is usually the first step of every task, where TextFlows employs various widgets to enable loading document corpora, labeling of documents with domain labels and converting them into the *AnnotatedDocumentCorpus* structure. We identified the following text document acquisition scenarios, which are also supported by the developed widgets:

- *Loading locally stored files in various application dependent formats.* In TextFlows document corpora can be uploaded from local files using the *Load Document Corpus* interactive widget. The entire dataset can be either a single text file (.doc, .docx, .pdf file formats are supported), where each line represents a separate document, or a zip of files in which a document is represented as a file. Apart from text, the files may optionally contain document titles, as well as multiple labels, which are encoded by the first word within a document prefixed by an exclamation mark, e.g., “!positive” is used to denote that the document belongs to the “positive” document category.
- *Acquiring documents using the WSDL+SOAP web services.* The user can integrate third-party web services as workflow components using the *Import webservice* button obtained from the bottom of the widget repository. Such integration allows for the inclusion of database web services into the workflow (e.g., PubMed offers a SOAP web service interface to access their database). TextFlows currently supports WSDL as the interface definition language for describing connections and requests to the web service and SOAP as the format for sending and receiving messages. The output of the imported web service widget can be connected to the *Load Document Corpus* widget that transforms the plain text input documents into the *Annotated-DocumentCorpus* structure.

- *Selecting documents from SQL databases.* The TextFlows platform supports loading data from MySQL databases via the *Load Document Corpus from MySQL* widget. Before execution the user enters the information which is required to connect to a database (e.g., user credentials, database address, database name, table name, column names, etc.) in order to retrieve the data from a MySQL database server. This widget then connects to the specified MySQL database server and returns the input columns representing document titles, texts and labels from the selected table. The final output of the widget is an automatically constructed *Annotated Document Corpus* object.

- *Crawling the Internet for gathering documents from web pages.* The *Crawl URL links* widget receives as an input a list of URLs, e.g., Wikipedia pages. First, every page is visited by an automatic crawler in order to gather the website's (HTML) content. Next, the Boilerpipe library [87] is used to extract the linguistically relevant textual content from the web page source. There are several content extraction methods available which can be selected by the user from the widget's properties. Finally, the *Crawl URL links* outputs the *Annotated Document Corpus* where document titles are represented with URLs and the extracted website texts become the document texts.

- *Collecting documents from snippets returned from web search engines.* In TextFlows the user can search the web using the *Search with Bing* and *Search with Faroo* widgets, which use the Microsoft Bing[11] and Faroo[12] as their web search engines, respectively. Both widgets require the user to enter the search query as well as the number of search results that the widget should return. The output of both widgets is a list of URLs which are returned by the web search engine. After the execution the output can be connected to the *Crawl URL links* widget which will extract the web content for every URL.

The most straightforward technique to incorporate background knowledge about the documents and their domain is to use a controlled vocabulary. A controlled vocabulary is a lexicon of all terms that are relevant for a given domain. TextFlows allows the users not only to upload their own local vocabulary files, but also gives them the possibility to use one of the implemented vocabulary construction tools, such as the *MeSH filter* widget, which constructs a vocabulary containing all the terms that belong to the user selected descriptors from the MeSH hierarchy[13].

3.4.2 Corpus Manipulation and Visualization

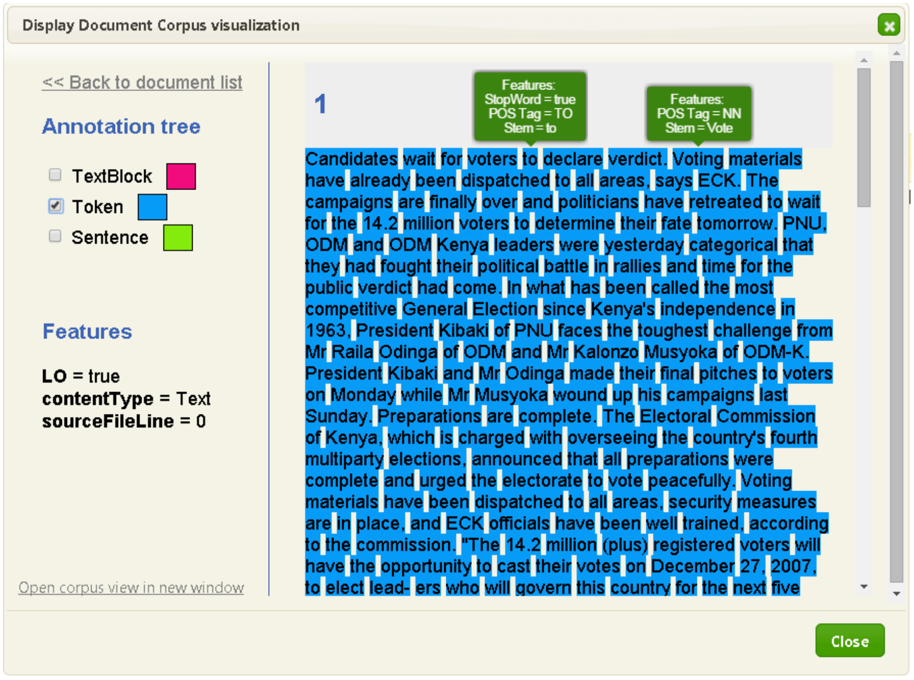
TextFlows implements widgets that allow the manipulation of *AnnotatedDocumentCorpus* (ADC) data objects. They allow the user to add new features, extract existing features from a document corpus, split document corpora (by either specifying conditions or by indices), merge different corpora, etc.

A special widget in the platform is *Document Corpus Viewer*, which visualizes the ADC data objects (note that TextFlows emphasizes the importance of the common ADC representation which is passed among the majority of widgets). As shown in Figure 3.3, the *Document Corpus Viewer* interactive widget allows the user to check the results of

[11]`http://www.bing.com/`

[12]`http://www.faroo.com/`

[13]MeSH (Medical Subject Headings) is a controlled vocabulary thesaurus used for indexing articles in PubMed, a database designed by The National Library of Medicine. The MeSH database is available online `http://www.nlm.nih.gov/mesh`



Display Document Corpus visualization

<< Back to document list

Annotation tree

☐ TextBlock 🟥
☑ Token 🟦
☐ Sentence 🟩

Features

LO = true
contentType = Text
sourceFileLine = 0

Open corpus view in new window

1

Features: StopWord = true POS Tag = TO Stem = to

Features: POS Tag = NN Stem = Vote

Candidates wait for voters to declare verdict. Voting materials have already been dispatched to all areas, says ECK. The campaigns are finally over and politicians have retreated to wait for the 14.2 million voters to determine their fate tomorrow. PNU, ODM and ODM Kenya leaders were yesterday categorical that they had fought their political battle in rallies and time for the public verdict had come. In what has been called the most competitive General Election since Kenya's independence in 1963, President Kibaki of PNU faces the toughest challenge from Mr Raila Odinga of ODM and Mr Kalonzo Musyoka of ODM-K. President Kibaki and Mr Odinga made their final pitches to voters on Monday while Mr Musyoka wound up his campaigns last Sunday. Preparations are complete. The Electoral Commission of Kenya, which is charged with overseeing the country's fourth multiparty elections, announced that all preparations were complete and urged the electorate to vote peacefully. Voting materials have been dispatched to all areas, security measures are in place, and ECK officials have been well trained, according to the commission. "The 14.2 million (plus) registered voters will have the opportunity to cast their votes on December 27, 2007, to elect lead- ers who will govern this country for the next five

Close

Figure 3.3: Document Corpus Viewer widget visualizes the *AnnotatedDocumentCorpus* data objects. This figure shows the document exploration page which displays a detailed view for a selected document. On the left side a list of applied tokenizations is shown, while on the right side the document's full text is displayed. When the user selects a tokenization from the list, the tokens are highlighted directly on the text of the document with an alternative background color. As shown in the figure, annotation features are shown by hovering over a token.

individual widgets by visualizing the ADC data object from their outputs. Through its interactive interface the widget allows the user to select an individual document from the list of document snippets by simply clicking on it. This opens a new page with a detailed view of the selected document, as shown in Figure 3.3.

3.4.3 Text Preprocessing Widgets

Preprocessing is a very important part of any form of knowledge extraction from text documents. Its main task is the transformation of unstructured data from text documents into a predefined well-structured data representation. In general, the task of preprocessing is to construct a set of relevant features from text documents. The set of all features selected for a given document collection is called a representational model [1], [88], [89]. Each document is represented by a vector of numerical values, one for each feature of the selected representational model. Using this construction, we get the most standard text mining document representation, called feature vectors, where each numeric component of a vector is related to a feature and represents a weight related to the importance of the feature in the selected document. Typically, such text-based feature vectors are very sparse, as the majority of weights are equal to zero [1]. The goal of text preprocessing widgets is to extract a high quality feature vector for every document in a given document corpus.

Technically, our implementation employs the LATINO (Link Analysis and Text Mining Toolbox) software library [9], NLTK (Natural Language Toolkit) library [10] and scikit-learn library [11] for the text preprocessing needs. Together they contain the majority of elementary text preprocessing procedures as well as a large number of advanced procedures that support the conversion of a document corpus into a table of instances, thus converting every document into a table row representation of an instance.

In the TextFlows widget repository preprocessing techniques are based on standard text mining concepts [1] and are implemented as separate categories. Every category possesses a unique hub widget, which has the task of applying a given preprocessing technique from its category to the *AnnotatedDocumentCorpus* data object. Every such widget is library independent, meaning that it can execute objects from either LATINO, NTLK or scikit-learn libraries.

A standard collection of preprocessing techniques is listed below, together with sets of functionalities implemented in our platform:

- *Tokenization.* In tokenization, meaningful tokens are identified in the character stream of the document, such as words or terms. TextFlows offers a large set of tokenizers: from LATINO, such as *Max Entropy Tokenizer* (word and sentence), *Unicode, Simple* and *Regex* tokenizers; various tokenizers from the NLTK library, from simpler ones, such as *Line, Regex, S-Expression, Simple*, to more complex ones, like *Stanford Tokenizer* and *Treebank Word Tokenizer.* Every tokenizer can be applied on a document corpus using the *Tokenizer Hub* widget. This hub receives as an input an ADC data object and a tokenizer instance, as well as two parameters entered by the user: the type of annotations to be tokenized (e.g., "TextBlock") and the type of annotations to be produced (e.g., "Sentence", "Token"). The *Tokenizer Hub* finds annotations of the input type and tokenizes them using the input tokenizer. The output of the hub is a new ADC object, which now contains the annotations of the new type. As described in the previous section, the results of any corpus tokenization can be visualized using the *Display Document Corpus* widget, as shown in Figure 3.3.

- *Stop word removal.* Stop words are predefined words from a language that do not carry relevant semantic information (e.g., articles, prepositions, conjunctions, etc.); the usual practice is to ignore them when building a feature set. In TextFlows we have three widgets which are used for stop word tagging: *Get StopWord Set* (outputs a predefined list of stop words for the user selected language—stop word lists for 18 languages, taken from Snowball[14], are included in our library), *StopWords Tagger* (receives as an input a list of stop words and outputs a constructed tagger object, which tags the words from the input list as stopwords), *StopWord Tagger Hub* (responsible for applying a stop word tagger on a document corpus). Similarly to the *Tokenization Hub*, the *Stop Word Tagger Hub* receives on its inputs an ADC data object and a stop word tagger instance. The user is able to enter two parameters: the type of annotations to be tagged (as a stop word) and a feature name, which is added (with a default value of 'true') to every annotation of the selected type, which the tagger marks as a stop word. The output of the hub is a new ADC object. Figure 3.3 shows the visualization of a selected document from the output ADC data object using the *Display Document Corpus* widget. The stop word annotation features are shown by hovering over the documents tokens.

- *Part-of-Speech tagging.* Tagging annotates words with the appropriate Part-of-Speech (PoS) tags based on the context in which they appear. The TextFlows platform

[14]Snowball: A small string processing language designed for creating stemming algorithms: `http://snowball.tartarus.org/`

includes the LATINO's *Max Entropy PoS Tagger* and from NLTK the following taggers: *Affix PoS Tagger* (learns only on term affixes), *Brill's rule-based PoS Tagger* [90], *Classifier-based PoS Tagger* (requires a classifier and a pre-annotated dataset to learn the PoS tags) and a *PoS N-gram tagger* (learns on a pre-annotated dataset the most frequent tag for every n-gram). PoS tags are applied to ADC data using the *PoS Tagger Hub*. The *PoS Tagger Hub* requires, besides the usual element annotation type (default: "Token") and the PoS feature name (default: "PoS Tag"), an additional parameter input by the user: a sentence annotation type (default: "Sentence"). The hub tags element annotations in the context of sentence annotations by assigning them new features with values returned by the PoS tagger. Figure 3.3 visualizes a selected document from the output ADC data object using the *Display Document Corpus* widget. The generated PoS tag features are shown when hovering over the document's tokens.

- *Stemming and lemmatization.* This is the process of converting words/tokens into their stem or citation forms. The following stemmers/lemmatizers were taken from the LATINO library: *Stem Tagger Snowball* and the *Lemma Tagger LemmaGen* [91]. We have also implemented the following widgets which represent the corresponding algorithms from the NLTK library: *Porter Stemmer*, *Snowball Stemmmer*, *ISRI Stemmer*, *Regex Stemmer*, *RSLP Stemmer*, *Lancaster Stemmer*, and *WordNet Lemmatizer*. Analogous as in the stop word removal category, stemmers and lemmatizers can be applied using the *Stem/Lemma Tagger hub*. This widget receives as an input an ADC data object and a stemmer (or lematizer) instance and outputs a new ADC data object with an additional stemming added. The user can enter two parameters: the type of annotations to be stemmed ("Token" by default) and a feature name ("Stem" by default), which will be assigned to every annotation of the selected type as a key-value pair together with the stemmed value.

3.4.4 Bag-of-Words Model

In the most general case, when dealing with raw text, the features are derived from text using only text preprocessing methods. The most common document representation model in text mining is the Bag-of-Words (BoW) model [1]. It uses all words (or, e.g., terms) as features, therefore the dimension of the feature space is equal to the number of different words in all of the documents. One of the most important characteristics of the described document features is the fact that they are usually very sparse [1], meaning that most of the features in a vector have zero weight. This sparsity is due to the fact that there are many different features (words, terms, concepts) in the document corpus; yet, a single document contains only a small subset of them. Consequently, the resulting feature matrix will have many (typically more than 99%) feature values that are zeros [1].

The TextFlows platform uses the Compressed Sparse Row (CSR) matrices, implemented in the scipy.sparse package[15] in order to be able to efficiently store the matrix of features in memory and also to speed up algebraic operations on vectors and matrices. The CSR matrices make no assumptions about the sparsity structure of the matrix, and do not store any unnecessary elements. Their main purpose is to put the subsequent non-zeros of the matrix rows in contiguous memory locations. Usually three vectors are created: one for storing floating-point numbers (*values*), and the other two for integers (*col_ ind, row_ brk*). The values vector stores the values of the non-zero elements of the matrix, as they occur if reading through the matrix row by row. The *col_ ind* vector stores the column indexes of the elements in the val vector, while the *row_ brk* vector stores the locations in the *values*

[15]`http://docs.scipy.org/doc/scipy/reference/sparse.html`

vector that start a new row in the dense original matrix. The storage savings using this approach are significant. Instead of storing $m*n$ elements, we only use $2*\text{nnz}+m$ storage locations, where m is the number of rows, n is the number of columns and nnz is the number of non-zeros in the dense matrix.

In the data mining modeling phase (i.e. document classification or text clustering), each document from the ADC structure needs to be represented as a set of document features it contains. In TextFlows the *Construct BoW Dataset and BoW Model Constructor* widget takes as an input an ADC data object and generates a sparse BoW model dataset (which can then be handed to a classifier). In addition, the widget takes as an input several user defined parameters, which are taken into account when building the feature dataset:

- *Token Annotation.* This is the type of *Annotation* instances marking parts of the document (e.g., words, sentences or terms), which will be used for generating the vocabulary and the dataset.

- *Feature Name.* If present, the model will be constructed out of annotations' feature values instead of document text. For example, this is useful when we wish to build the BoW model using stems instead of the original word forms.

- *Stop Word Feature Name.* This is an annotation feature name which is used to tag tokens as stop words. These tokens will be excluded from the BoW representational model. If the stop word feature name in not provided, all tokens are included in the BoW space.

- *Label Document Feature Name.* This is the name of the document feature which will be used as a class label of the examples in the dataset. If blank, the generated sparse dataset will be unlabeled.

- *Maximum n-gram Length.* The upper bound of the range of n-values for different n-grams to be extracted. All values of n such that $1 \leq n \leq \text{maxNgram}$ will be used, where maxNgram is the maximum n-gram length.

- *Minimum Word Frequency.* Cut-off frequency value for including an item into the vocabulary.

- *Word Weighting Type.* The user can select among various weighting models for assigning weights to features:

  - Binary. The feature weight is 1 if the corresponding term is present in the document, or zero otherwise.
  - Term occurrence. The feature weight is equal to the number of occurrences of the corresponding term. This weight is sometimes better than a simple binary value since frequently occurring terms are likely to be more relevant for the given text.
  - Term frequency. The weight is derived from the term occurrence by dividing the vector by the sum of all vector's weights.
  - TF-IDF. Term Frequency-Inverse Document Frequency [88] is the most common scheme for weighting features. For a given term w in document d from corpus D, the TF-IDF measure is defined as follows:

$$\text{tfIdf}(w,d) = \text{tf}(w,d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{3.1}$$

where $\text{tf}(w, d)$ represents the number of times term w appears in document d. The reasoning behind the TF-IDF measure is to lower the weight of terms that appear in many documents as this is usually an indication of them being less important (e.g., stop-words). The appropriateness of this scheme was confirmed in numerous text mining problem solutions [1], [92].

– Safe TF-IDF. For a given term w in document d from corpus D, the Safe TF-IDF measure is defined as follows:

$$\text{safeTfIdf}(w, d) = \text{tf}(w, d) \times \log \frac{|D|}{|\{d \in D : w \in d\}| + 1}, \tag{3.2}$$

This approach smoothens IDF weights by adding one to document frequencies, as if an extra document was observed containing every term in the collection exactly once. This prevents the occurrence of divisions by zero.

– TF-IDF with sublinear TF scaling. It often seems unlikely that twenty occurrences of a term in a document truly carry twenty times the significance of a single occurrence. Accordingly, there has been considerable research into variants of term frequency that go beyond counting the number of occurrences of a term [93]. A common modification is to use the logarithm of the term frequency instead of *tf*, defined as:

$$\text{wf}(w, d) = \begin{cases} 1 + \log \text{tf}(w, d), & \text{if } \text{tf}(w, d) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

- *Normalize Vectors.* The weighting methods can be further modified by vector normalization. If the user opts to use it in TextFlows, the $L2$ regularization [94] is performed.

Besides the sparse BoW model dataset the *Construct BoW Dataset and BoW Model Constructor* also outputs a *BowModelConstructor* instance. This additional object contains settings which allow repetition of the feature construction steps on another document corpus. These settings include the input parameters, as well as the learned term weights and vocabulary.

An important widget in the Bag-of-Words category is the *Create BoW Dataset using BoW Model Constructor*. Its task is to apply the input *BowModelConstructor* instance to an input ADC data object and create a sparse feature dataset. This is useful, for instance, in every cross-validation fold where you need to build the test dataset's sparse matrix using the same settings (also including IDF term weights) used for building the training sparse matrix.

3.4.5 Document Classification

A document classification (also called text categorization) refers to automated assignment of predefined categories to natural language texts. A primary application of text categorization systems is to assign subject categories to documents to support information retrieval, or to aid human indexers in assigning such categories. Text categorization components are also increasingly used in natural language processing systems for data extraction. Classification techniques have been applied to spam filtering, language identification, genre classification, sentiment analysis, etc. The common approach to building a text classifier is to manually label a selected set of documents to predefined categories or classes, and use them to train a classifier. The trained classifier can then be used to assign class labels to unseen documents based on the words they contain.

The term *supervised learning* refers to the above-described approach to automatically building a text classifier from training documents, which have been labeled (often manually) with predefined classes. The TextFlows platform currently contains only the supervised learning approaches from the LATINO, NTLK and scikit-learn libraries. Every widget contains input parameters which are used to construct the classifier object. From the LATINO library we integrated *Nearest Centroid Classifier, Naive Bayes Classifier, SVM (Binary and Multiclass) Classifier, Majority Classifier, Maximum Entropy Classifier, kNN (fast and regular version) Classifier*, while the NLTK library contributes an additional Naive Bayes Classifier. The following widgets represent classifiers from the scikit-learn library: *Decision Tree Classifier, Multinomial Naive Bayes Classifier, Gaussian Naive Bayes Classifier, k-Nearest Neighbours Classifier, SVM Linear Classifier, SVM Classifier.*

For training and applying classifiers TextFlows offers two dedicated widgets: *Train Classifier Hub* and *Apply Classifier Hub*. The *Train Classifier Hub* receives on its inputs a sparse feature dataset object and an untrained classifier. Its function is to fit the classifier model according to the given training data. The final outcome of this widget is a trained classifier object.

The *Apply Classifier Hub* receives a trained classifier object and returns predicted class probabilities for every new document from the input test dataset, as well as the test dataset with a new predicted label column.

3.4.6 Literature-Based Discovery

This category of widgets supports the literature-based discovery process. The package contains several widgets which specify different elementary heuristics. These basic heuristics are grouped into one of five categories: frequency-based, TF-IDF-based, similarity-based, outlier-based, banded matrix-based. The first four categories are an implementation of the heuristics proposed by Juršič et al. [17], [70], while banded matrix-based heuristics are newly proposed and are presented in more detail in Section 5.4. Each category is represented by an interactive widget, which allows the user to manually select its elementary heuristics through an interactive dialog. The literature-based discovery package also contains several widgets that specify operations between elementary heuristics, such as minimum, maximum, sum, norm, etc., which can be used for building complex ensembles, as we show in Section 5.3.

The library also contains two widgets that support the specification of ensemble heuristics, which is described in Section 5.2.3: *Ensemble Heuristic Vote* and *Ensemble Average Position* widget. The first defines an ensemble voting heuristic (it calculates term votes according to Equation 5.1 of Section 5.2.3), while the latter specifies an ensemble that calculates normalized sum of term position scores of the input heuristics (see Equation 5.2 of Section 5.2.3).

The most important widget from this package is the *Calculate Term Heuristic Scores* widget which takes as an input several heuristics specifications and performs the actual calculations. The decision for such an approach—having one widget which calculates all the heuristics—is that several elementary heuristics require the same intermediate results. These results can be cached and calculated only once, which results in faster computation. To this end, the TextFlows platform uses Compressed Sparse Row (CSR) matrices[16] to be able to store the matrix of features in memory and also to speed up algebraic operations on vectors and matrices. The *Calculate Term Heuristic Scores* widget also takes as input the *BowModelContructor* object and the *AnnotatedDocumentCorpus*. The parse settings from

[16]Compressed Sparse Row (CSR) matrices are implemented in the scipy.sparse package `http://docs.scipy.org/doc/scipy/reference/sparse.html`

the *BowModelConstructor* object are used to construct Compressed Sparse Row (CSR) matrices, which represents the BoW model. TextFlows uses mathematical libraries numpy and scipy to efficiently perform the heuristics calculations.

Literature-based discovery package also contains the *Explore in CrossBee* widget which exports the final ranking results and the annotated document corpus into web application CrossBee, which offers manual exploration of terms and documents. Also, the *Rank Terms* widget can be used to display the ranked terms in the form of a table along with their respective scores.

3.4.7 Noise Detection

Noise filtering is frequently used in data preprocessing to improve the accuracy of induced classifiers. TextFlows incorporates an ensemble-based noise ranking methodology for explicit noise and outlier identification, named NoiseRank [95], which was modified to work with texts and TextFlows ADC data objects. Its main aim is to detect noisy instances for improved data understanding, data cleaning and outlier identification. NoiseRank was previously successfully applied to a real-life medical problem [95]. We show an example of using the NoiseRank methodology on a task of outlier detection in document corpora in Section 4.5.

3.4.8 Evaluation and Visualization

The TextFlows platform enables users to create interactive charts for easy and intuitive evaluation of performance results. It includes standard performance visualizations used in machine learning, data mining, information retrieval, etc. Notably, the TextFlows platform includes a full methodology, named VIPER [95], [96], a visualization approach that displays the results as points in the two-dimensional precision-recall space. The platform contains several visual performance evaluation widgets, which result in interactive performance charts that can be saved and exported to several formats.

- *Scatter charts.* These include ROC space charts and PR space charts.
- *Curve charts.* These include PR curves, Cost curves, Lift curves, ROC curves, Kendall curves and Rate-driven curves.
- *Column charts.* These are general column charts for visualizing multiple performance measures for a set of algorithms.

While VIPER visualizations appear to be straightforward, this visualization toolbox is innovative and very useful for text analytics. An example is visual comparison of F-value results of different text analysis tools, including F-isoline-based text classifier comparison, which is not supported by any other visualization tool. We demonstrate several implemented visualization techniques in Section 4.2.

3.4.9 Inductive Logic Programming

The current Inductive logic programming (ILP) module includes a propositionalization technique named *wordification* [97], [98] which can be seen as a transformation of a relational database into a corpus of text documents. Wordification constructs simple, easy to understand features, acting as words in the transformed Bag-of-Words representation. The main advantages of the approach are: simple implementation, accuracy comparable to competitive methods, and greater scalability. We describe the wordification methodology in more detail and compare it to several propositionalization techniques in Chapter 6.

The ILP module also includes components, such as the popular ILP system Aleph [71], as well as RSD [75], SDM [99] and RelF [76]. Aleph is an ILP toolkit on its own, with a wide range of functionalities: from decision tree learning to feature generation and first-order rule induction. Relational Subgroup Discovery (RSD) algorithm implements a propositionalization approach. It starts with a typical relational ILP domain and converts it into a single-table representation; this is done by generating a set of first-order features which become attributes of the propositionalized training examples. Although RSD comes with its own implementation of the CN2-SD [100] subgroup discovery algorithm, the resulting table can serve an input into any propositional machine learning or data mining algorithm.

3.5 Comparison with Other Text Mining Platforms

In Section 2.2 we presented an overview of more widely used workflow management systems, with a focus on those that are also or primarily used for NLP and that support distributed and/or remote processing. In the following sections we compare the overviewed systems with TextFlows along several dimensions that affect the usefulness of each system.

3.5.1 Open Source

The first dimension, summarized in Table 3.1, concerns the question whether the workflow management system is open source, i.e. whether it is possible to download the complete system and install it on local server(s). This is important in cases where the system is to be an internal one, not accessible to third parties, e.g., for data privacy protection or where local modifications to the system are desired. In the general case, this option is not really needed, as it is much easier to simply use the official system. Nevertheless, it is desirable to at least have this option available. Of the surveyed systems, Taverna is available under OS. It can be installed on all major platforms (Windows, Mac, Linux) with precompiled distributions or in source Java and has few prerequisites; Language Grid (available from SourceForge) also runs on all platforms and requires PostgreSQL and Tomcat, with the set-up being rather complicated. TextFlows is also open source and publicly available to download and install; all the prerequisites are wrapped in the installation script.

Table 3.1: Comparison of NLP platforms regarding open source.

Taverna	Yes (Java)
PANACEA	No
Argo	No
WebLicht	No
Language Grid	Yes (Java, PostgreSQL, Tomcat)
TextFlows	Yes (Python)

3.5.2 Workflow Sharing

All systems offer workflow sharing, as this is the essence of building such systems, however, they differ in whether the platform itself provides the registry and exploration service of registries or whether they make use of associated services for workflow sharing and discovery. As can be seen from Table 3.2, most systems provide their own sharing platform, with the exception of Taverna and PANACEA, both of which use myExperiment as a social network platform and BioCatalogue (and, in the case of Taverna, other catalogues as well) as a registry for public web services.

Table 3.2: Comparison of NLP platforms regarding workflow sharing.

Taverna	With myExperiment, BioCatalogue, etc.
PANACEA	Own installation of BioCatalogue and MyExperiment
Argo	Local registry
WebLicht	Local registry
Language Grid	Local registry
TextFlows	Local registry

3.5.3 Simplicity of Use

This dimension, summarized in Table 3.3, describes how difficult it is to start using a particular system, not so much from the point of view of executing ready-made workflows but of designing workflows and adding new web services to the platforms. Taverna provides a dedicated workflow management system, which is, however, mostly due to the variety of options, reportedly quite difficult to master, also because the details of web service communication have to be understood by the user and complex types decomposed into atomic parts. Furthermore, the different third-party web services are often not compatible. It is also possible to add new web services using SoapLab. The composition of workflows in PANACEA follows that of Taverna, so the same (dis)advantages apply. Argo provides a web-based interface, which allows for graphical composition of workflows. It also provides a sophisticated system for adding new web services, including a testing environment. WebLicht supports web-based workflow composition, with the registry constraining which web services can be attached to the workflow, depending on their I/O requirements. WebLicht also offers a good document viewer and a very compatible set of components. However, new web services cannot be added by users. Language Grid does enable composition of new workflows, but this is an off-line and rather complicated process without a visual editor. Finally, TextFlows offers, via a web interface, easy composition of workflows. Adding new native Python algorithms as well as web SOAP-based services with WSDL available is also trivial (based on the user-supplied URL of WSDL, new workflow components are created automatically from service's functions).

Table 3.3: Comparison of NLP platforms regarding simplicity of use.

Taverna	Difficult
PANACEA	Difficult
Argo	Easy for workflow composition
WebLicht	Easy for workflow composition
Language Grid	Difficult
TextFlows	Easy for workflow composition

3.5.4 I/O Protocols

This dimension, summarized in Table 3.4, concerns input/output protocols. The options supported by the communication protocols between the web services in a workflow mostly depend on the age of the system: the older ones typically prefer WSDL and SOAP, while the newer ones choose the simpler REST.[17]

Table 3.4: Comparison of NLP platforms regarding I/O protocols.

Taverna	WSDL+SOAP, REST
Argo	REST
PANACEA	WSDL+SOAP
WebLicht	REST
Language Grid	WSDL+SOAP
TextFlows	WSDL+SOAP, REST, JSON-WSP

When dealing with NLP workflows, standards become very important. NLP components mainly perform pipeline processing, where each tool/service adds another layer of annotation to the base text where it typically needs to have access to previous annotations. Furthermore, the input text might itself contain annotations, such as document structure. The wish for interoperability leads to the development of standards for text (corpus) annotation; however there are a number of such standards and best practices, and different systems use different ones. TAVERNA, not being targeted towards NLP, is standard agnostic, and relies on implementers of workflows to provide the necessary conversion (shiv) services, i.e. conversion routines to the format that the downstream services expect. In the context of NLP it is also possible to develop shiv services that convert not only outputs but also inputs, taking one of the commonly accepted standards as the pivot encoding. This is the route taken by PANACEA, where each web service is wrapped to accept and produce an output: for primary data this is an XCES encoded file, while text annotations use the LAF/GrAF [101] standard with stand-off annotations. Argo is based on UIMA[18] and does not enforce any NLP standards in its pipelines. However, Argo did pioneer a new method of aligning I/O requirements of component web services: rather than relying on shiv services, which need programming skills, it supports a Type Mapper, i.e. a dedicated rule-based analytic for transcribing feature structures between types needed for particular web services. It also supports export and type mapping to RDF.

WebLicht uses a local format TCF, which is, however, quite similar to ISO LAF/GrAF and conversion to these is provided. In WebLicht all web services are expected to use TCF, where the conversion is typically implemented as a wrapper around each annotation tool. Furthermore, the WebLicht Registry included information about the prerequisites for each web service and allows only chaining web services that do not violate these constraints. Language Grid has a very different scheme from most other services, and the details of how the interfaces are to be configured are rather hard to come by. In general, it defines an ontology of web services, which then specifies also their I/O formats. As the focus is on dictionaries and machine translation, it is these kinds of formats that are defined. Finally, TextFlows, as Taverna, does not impose any standards in its I/O formats. Mostly plain text, JSON and, for efficiency, serialized Python data structures and objects (in particular, the *AnnotatedDocumentCorpus* instances) are exchanged between workflow components.

[17]Note that the WSDL protocol referred to here is version 1.0 or 1.1; WSDL 2.0 offers (limited) support for RESTful web services.

[18]`https://uima.apache.org/`

3.5.5 Popularity and Functionality

The dimension, summarized in Table 3.5, concerns the question of popularity and functionality of the presented workflow management systems. We take several factors into consideration, such as the size of the user base, number of text mining and NLP widgets available, number of workflows shared, etc. We grouped the user base into three categories: 'Large' (over 100,000 users), 'Medium' (between 100 and 100,000 users) and 'Small' (less than 100 users). Similarly, we also grouped the number of workflows, where 'Large' represents more than 1,000 publicly available workflows, 'Medium' between 100 and 1,000 workflows and 'Small' less than 100. In the case regarding the number of NLP widgets 'Large' represents more than 80 NLP components, 'Medium' between 30 and 80, while 'Small' measures less than 30 NLP components.

Table 3.5: Comparison of NLP platforms regarding the extent of their popularity and functionality.

Platform	User base	Number of workflows	Number of NLP components
Taverna	Large	Large	Small
Argo	Small	Small	Small
PANACEA	Small	Small	Small
WebLicht	Medium	Medium, oriented towards corpus processing	Large for German and English
Language Grid	Large	Large, application oriented tools	Large for Japanese, medium for English
TextFlows	Small	Small	Large for English

Out of the surveyed systems, Taverna is the most developed and the most popular platform, with many features and connection capabilities. The myExperiment website currently contains over 2,000 Taverna workflows, but mostly from bioinformatics and other life science domains—there are only a few NLP workflows available. Furthermore, Taverna contains only a few simple widgets, which are tightly integrated into the platform, the rest must be added as web services. On the other hand, Argo is a relatively new platform, which is still in its beta development phase. Consequently, it has a much smaller user base and does not yet offer construction and exploitation of publicly shared workflows. It offers a relatively large set of bioinformatic widgets, as well as some NLP tools. With the PANACEA platform there does not seem to be any great up-take of this service by the NLP community. The inspection of the PANACEA's MyExperiment repository[19] shows that the platform offers only a few workflows or web services. It implements only a few basic NLP tools for more or less 5 languages and most of these are fairly specific, developed by the (ex) project partners. As the most recent workflow was published in 2013, and the fact that at the time of writing this thesis the platform was not available online, it seems highly unlikely that this platform will ever be adopted by the NLP community. Language Grid is an open source service based framework which enables access to various language services in the world based on a single powerful protocol HTTP. Therefore, it is widely popular and also contains a large number of NLP services for various languages. On the other hand, the WebLich platform is not open source, which might contribute to a lower popularity compared to Teverna or Language Grid. Nonetheless, it is oriented towards corpus processing, so it offers a large number of NLP components. TextFlows is

[19]`http://myexperiment.elda.org/workflows`

the newest of the presented platforms, hence it has the smallest user base and number of publicly shared workflow of all presented platforms. It does however possess the largest set of text mining and NLP widgets among the presented platforms, but mainly suited for English corpora.

3.6 Comparison with ClowdFlows

TextFlows is a heavily modified fork of the ClowdFlows [12] data mining platform, which is a general workflow construction and execution tool for data mining and machine learning. The platform incorporates workflow components for numerous data mining and machine learning projects—including WEKA [29] and Orange [30]—while severely lacking text mining and natural language processing components. Moreover, despite being very useful for workflow construction by the informed developers and end-users, ClowdFlows currently suffers from a somewhat disorganized roster of workflow components which may be incompatible with each other. As a result, new users sometimes struggle to construct functioning workflows as there are too many mutually incompatible components.

We created a fork of ClowdFlows in order to maintain cohesion of incorporated text mining and natural language processing workflow components. TextFlows has a completely redesigned roster of workflow components, which are now highly compatible with each other within the platform and easier to use for the expert and novice users. In contrast to ClowdFlows, where there are no guidelines for sorting workflow components into categories, TextFlows sorts the components based on the functionality of the components. As a result, all widgets that perform similar functions are located in the same category. We have also enriched the user interface with extra information about the workflow components and written the documentation for the provided workflow components and their inputs and outputs.

While we introduced a completely new common text representation structure (cf. Section 3.2.2), a new widget structure as well as numerous new text mining and new NLP components (cf. Section 3.4) and workflows (cf. Chapter 4 for selected examples), the underlying architectures of ClowdFlows and TextFlows remain similar. To summarize, TextFlows is built on top of ClowdFlows, meaning that both the widget execution engine and the core of the ClowdFlows user interface are present in TextFlows. TextFlows still benefits from all security updates, bug fixes and feature upgrades that ClowdFlows receives.

Chapter 4

Text Mining and Natural Language Processing Use Cases

In this chapter we address novel natural language processing and knowledge discovery use cases, which rely on the TextFlows implementation described in the previous chapter. The main goal of this chapter is to ultimately justify the development of a web-based, cloud-based knowledge discovery platform for text mining and NLP processing. For the natural language processing scenario, we demonstrate the advanced features of the developed platform on five use cases: a comparison of classifiers from different libraries for a text classification problem, a comparison of POS taggers on the same text categorization problem, a comparison of stemmers/lemmatizers from different libraries and outlier detection on document corpora.

In Section 4.1 we propose a document acquisition and text preprocessing workflow, which will also be either partly or fully used as a subprocess in other use cases. Furthermore, we also describe the Kenyan elections dataset [15], [16], which is the dataset used in the experiments and workflows proposed in this chapter. Next, a comparison of different classifiers from different libraries for a text classification problem is shown in Section 4.2, while Section 4.3 shows a comparison of several stemmers/lemmatizers on the same text categorization problem. Similarly, Section 4.4 proposes a workflow for comparison of POS taggers from different software libraries. Finally, we present an approach for outlier detection on document corpora as a workflow in the TextFlows platform in Section 4.5.

4.1 Document Acquisition and Preprocessing Workflow

The two common steps in the three presented use cases are the *Document acquisition* and *Document preprocessing* steps. These two steps were implemented as subprocesses in the main workflows for the four use cases, illustrated in Figures 4.2, 4.7, 4.9 and 4.11.

In the remaining four workflows described in this chapter, *Document acquisition* is the first step of the methodology and is responsible for, first, loading the locally stored file (representing the document corpus), then labeling the documents with appropriate domain labels and finally converting them into the *AnnotatedDocumentCorpus* data object. In our experiments we used a corpus of documents, presented in [15] and [16], which was originally collected by the IPrA Research Center, University of Antwerp. The document corpus contains 464 articles (about 320,000 words) concerning Kenyan presidential and parliamentary elections, held on 27th December 2007, and the crisis following the elections. The documents originate from six different daily newspapers in English, covering the time period from 22nd December 2007 to 29th February 2008. Articles from the US and British press (The New York Times, The Washington, The Independent, The Times and Post)

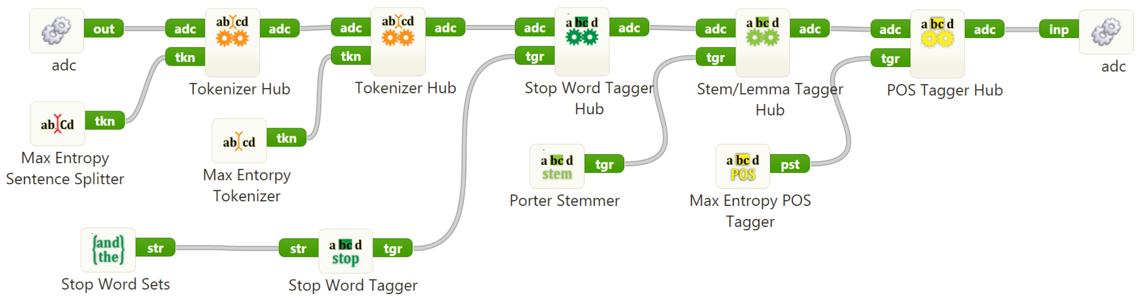


Figure 4.1: Document preprocessing workflow, which is publicly available at `http://textflows.perovsek.com/workflow/47/`. The same workflow is implemented as a subprocess in the four use cases presented in the rest of this chapter.

form the class label "Western" (WE) and articles from local Kenyan newspapers Daily Nation and The Standard are categorized as "Local" (LO).

Figure 4.1 shows the subprocess in the TextFlows platform for the second step in all four presented methodologies: the *Document preprocessing* workflow. As described in Section 3.4, category specific hubs are used for applying different preprocessing objects to the *AnnotatedDocumentCorpus* data object. The documents are first split into sentences with LATINO's *Max Entropy Sentence Splitter* and then the sentences are split into tokens with LATINO's *Max Entropy Tokenizer*. Some of these tokens are tagged as stop words using the *Stop Word Tagger* with the predefined Snowball list of English stop words. Next, the *Porter Stemmer* is used for converting tokens into their stems. Finally, the *Max Entropy PoS Tagger* is used to Part-of-Speech tag the input ADC data object.

4.2 Classifier Comparison for Text Categorization Workflow

In this section we propose a workflow for classifier evaluation, which is showcased on the Kenyan elections dataset. In our experiments we compared 7 different classifiers from different text mining libraries. As shown in Figure 4.2, we compared 4 classifiers from LATINO (*Nearest Centroid Classifier, Naive Bayes Classifier, Maximum Entropy Classifier, kNN Fast Classifier*) and 3 classifiers implemented in the scikit-learn library (*Gaussian Naive Bayes Classifier, k-Nearest Neighbours Classifier, SVM Linear Classifier*).

Every algorithm was evaluated using 10-fold stratified cross-validation, as shown in Figure 4.3. All cross-validations were performed using the same seed in order to ensure the data was equally split for different classifiers. Figure 4.4 shows the methodology behind the *CrossValidation* subprocess which was executed for every cross-validation fold. First, a sparse BoW model dataset and the *BowModelConstructor* instance were generated by the *Construct BoW Dataset and BoW Model Constructor*. We constructed unigrams (n-grams where n = 1) out of stemmed values from word tokens, while disregarding stop words in the process. The Bag-of-Words vector space was calculated using the TF-IDF weighting scheme.

The test sparse matrix is required to be constructed using the same parameter settings, IDFs and vocabulary as the training sparse matrix. Therefore, we applied the BoW constructor object (output of the *Construct BoW Dataset and BoW Model Constructor*) to the test ADC data object using the *Create BoW Dataset using the BoW Model Constructor*.

After calculating the training and testing sparse matrices, we fitted the input classifier model to the training dataset using the *Train Classifier Hub*. Next, the *Apply Classifier Hub* received the trained classifier object and returned predicted class probabilities for

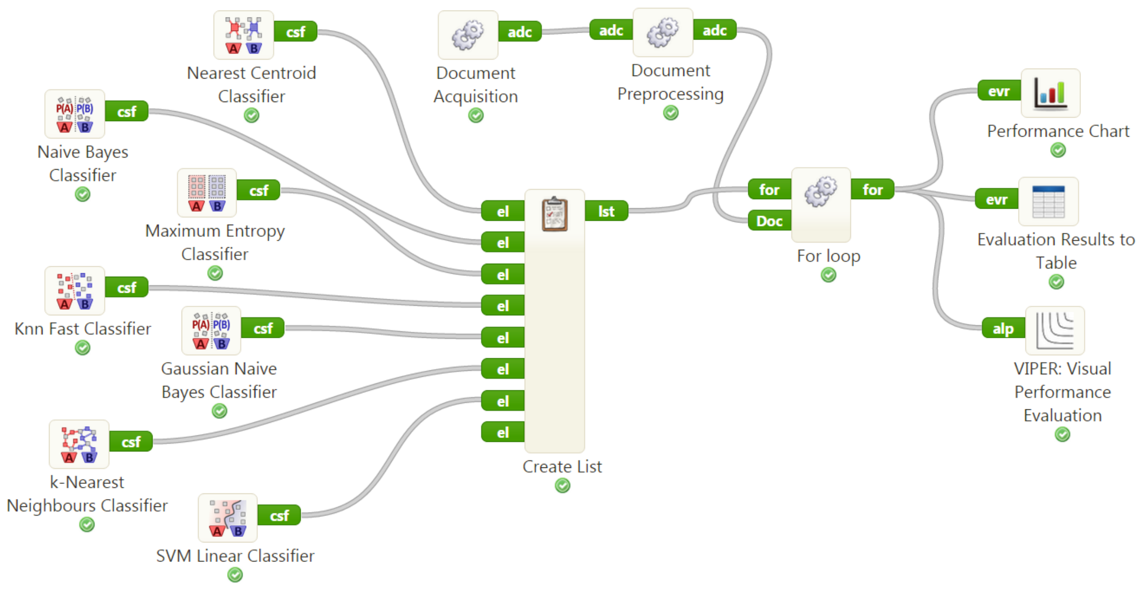


Figure 4.2: The workflow for evaluating 7 classifiers from different text mining libraries. The workflow is available at `http://textflows.perovsek.com/workflow/2/`.

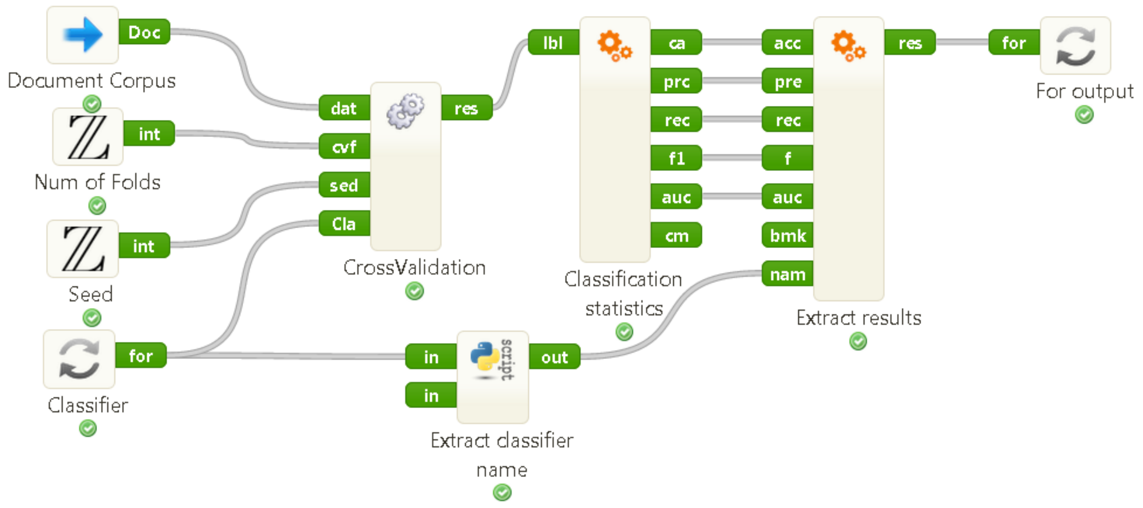


Figure 4.3: The *For loop* subprocess which evaluates the input classifier using 10-fold stratified cross-validation (with a constant seed) and extracts the obtained results, which are then visualized as shown in Figure 4.2.

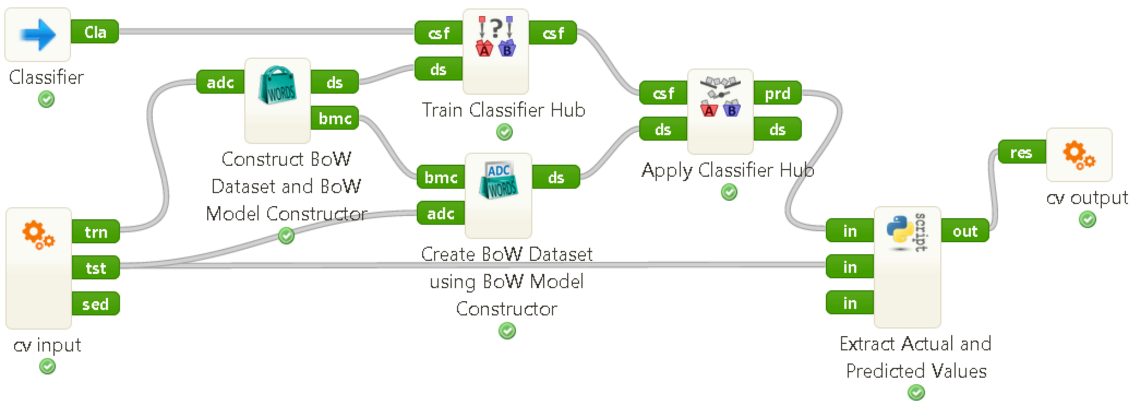

Cla
Classifier
adc
ds
bmc
Construct BoW Dataset and BoW Model Constructor
csf
ds
csf
Train Classifier Hub
bmc
adc
ds
Create BoW Dataset using BoW Model Constructor
csf
ds
prd
ds
Apply Classifier Hub
trn
tst
sed
cv input
in
in
in
out
Extract Actual and Predicted Values
res
cv output

Figure 4.4: The subprocess workflow representing the methodology in every fold for the used 10-fold cross-validation. First, the construction of the training and testing sparse matrices is performed. Next, the input classifier model is fitted to the training dataset and used to predict class probabilities for every new document from the input test dataset. The subprocess returns constructed pairs of actual and predicted classes.

every new document from the input test dataset. The *Extract Actual and Predicted Values* widget used these probabilities and constructed pairs of actual and predicted classes. These pairs were returned from the *CrossValidation* subprocess and used for calculating different metrics, as shown in Figure 4.3.

The results of cross-validation (precision, recall, F-score) were connected to the input of the *VIPER: Visual Performance Evaluation* widget, as shown in Figure 4.2, while Figure 4.5 presents its visualization of classifier evaluation in the precision-recall plane, where each point represents the result of an algorithm (for the selected class label “Local”). Points closer to the upper-right corner have higher precision and recall values. F-measure values are presented as isolines (contour lines), which allows a simple comparison of algorithm performance.

Figure 4.5 shows that in terms of F-measure, scikit-learn’s *SVM Linear Classifier* and LATINO’s *Maximum Entropy Classifier* achieved the best results: both algorithms generally achieved a higher percentage of correctly classified examples (higher recall score), and also a slightly higher percentage of correctly classified examples of the target class (higher precision score) compared to other classifiers used. A somewhat lower performance was achieved using LATINO’s *Nearest Centroid Classifier* classifiers, while the k-nearest neighbor and Naive Bayes classifiers performed worse. Detailed results are presented in Table 4.1.

Table 4.1: Classifier evaluation on the Kenyan elections dataset.

Library	Classifier	Recall	Prec.	F1 score	Classif. accuracy	AUC
LATINO	Nearest Centroid Classifier	0.96	0.90	0.93	92.42%	0.92
LATINO	Naive Bayes Classifier	1.00	0.72	0.84	80.74%	0.81
LATINO	Maximum Entropy Classifier	0.97	0.93	0.95	94.59%	0.95
LATINO	kNN Fast Classifier	0.93	0.88	0.90	90.26%	0.90
scikit-learn	Gaussian Naive Bayes Classifier	0.87	0.82	0.84	83.77%	0.84
scikit-learn	k-Nearest Neighbors Classifier	0.91	0.88	0.89	89.18%	0.89
scikit-learn	SVM Linear Classifier	0.95	0.95	0.95	95.24%	0.95

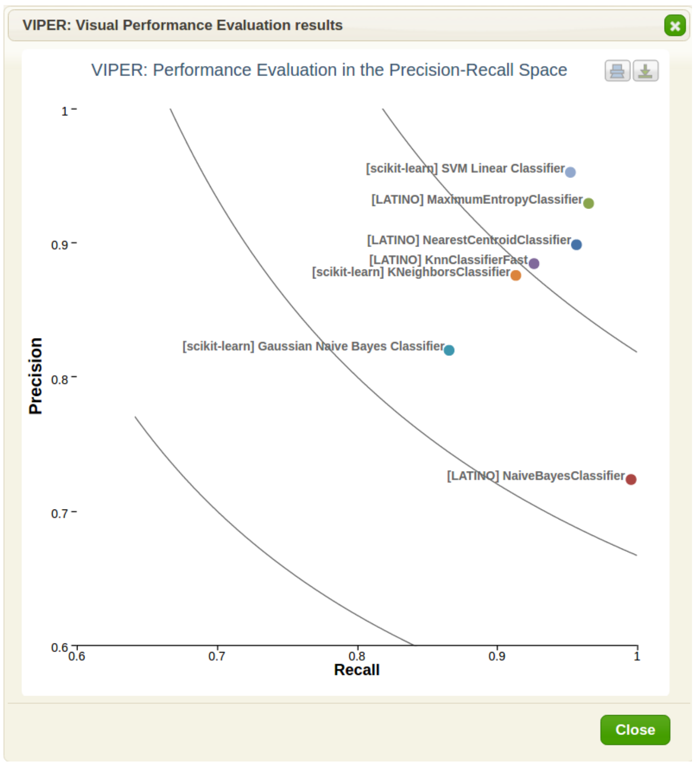


Figure 4.5: The VIPER visualization showing evaluation of classifiers from different libraries. This visualization is the result of the workflow presented in Figure 4.2.

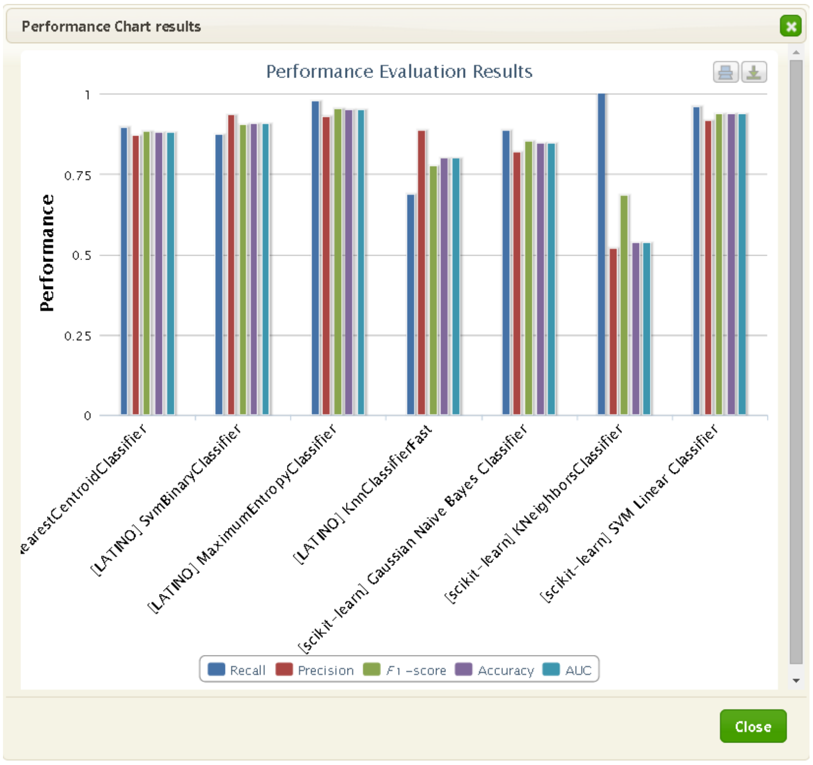


Figure 4.6: A column chart showing evaluation of classifiers from different libraries. This visualization is the result of the workflow presented in Figure 4.2.

The workflow was run within a virtual machine with a dedicated one core (Intel i7 2600k) and 4GB of RAM. The running time for the document acquisition and document preprocessing step is 161 seconds while the comparison of classifiers with the 10 fold cross-validation takes 1,486 seconds. The runtime of the entire workflow was 1,652 seconds.

4.3 Stemmer and Lemmatizer Comparison Workflow

This section demonstrates advanced features of TextFlows on a use case, comparing selected stemmers and lemmatizers on a text categorization problem.

The processing starts with the *Document acquisition* and *Document preprocessing* steps. These two steps were implemented as subprocesses in the complex stemmer/lemmatizer comparison workflow shown in Figure 4.7. As explained in Section 4.1, the *Document acquisition* is the first step of the methodology and is responsible for, first, loading the locally stored text file (containing the Kenyan elections document corpus), then labeling the documents with appropriate domain labels and finally converting them into the ADC data object. Next, the second step of the presented methodology—the *Document preprocessing* step—is performed. The documents are segmented into sentences with LATINO's *Max Entropy Sentence Splitter* and these tokenized using LATINO's *Max Entropy Tokenizer*. Some of the tokens are tagged as stop words using the *Stop Word Tagger* with a predefined list of English language stop words, taken from Snowball. Finally, the *Max Entropy PoS Tagger* is used to Part-of-Speech tag the input ADC data object. In this preprocessing subprocess we, exceptionally, omitted the application of a stemmer, as this was done later in the cross-validation subprocess.

In the experiments we compared six different stemmers: *RSLP Stemmer, Snowball Stemmer, ISRI Stemmer, Lancaster Stemmer* (from the NLTK library) and *Stem Tagger Snowball* (from LATINO library); as well as two lemmatizers: the NLTK *WordNet*

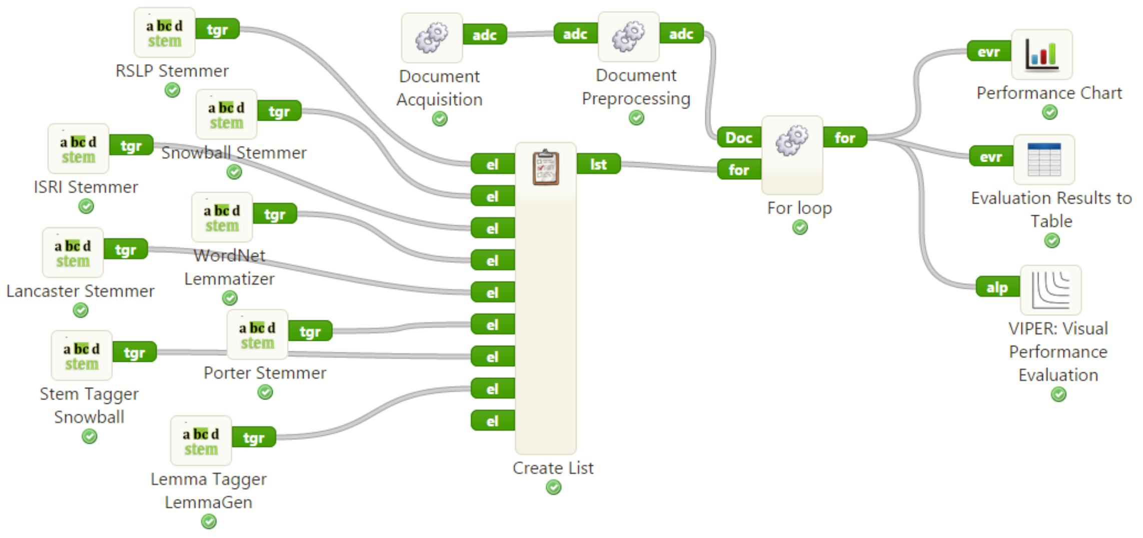


Figure 4.7: The workflow implemented in the TextFlows platform, used for the evaluation of six different stemmers and two lemmatizers, selected from various text mining libraries, on a text categorization problem. The abbreviations on the input and output stubs are as follows: *tgr* tagger, *el* element, *lst* list, *adc* annotated document corpus, *evr* and *alp* evaluation results. The workflow is publicly available at `http://textflows.perovsek.com/workflow/42/`.

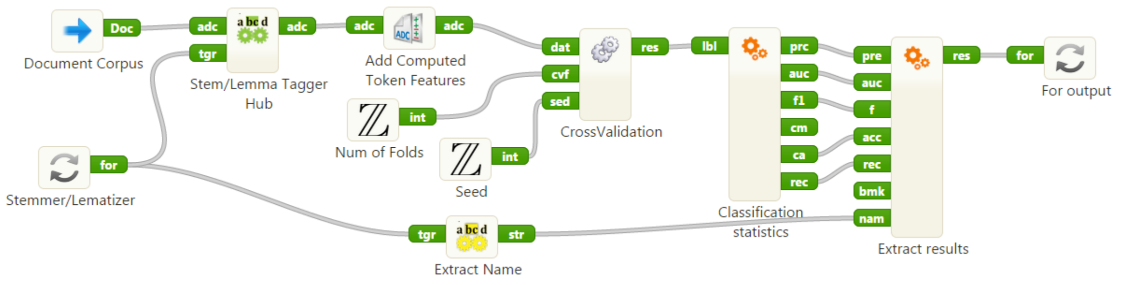


Figure 4.8: The subprocess workflow representing the methodology in every fold for the used 10-fold cross-validation. First, the input stemmer/lemmatizer is applied to the document corpus. Next, the construction of the training and testing sparse matrices is performed. Then, maximum entropy classifier is fitted to the training dataset and used to predict class probabilities for every new document from the input test dataset. Last, the subprocess returns constructed pairs of actual and predicted classes. The abbreviations on the input and output stubs are as follows: *doc* and *adc* annotated document corpus, *tgr* tagger, *adc* annotated document corpus *str* string, *int* integer, *cvf* number of cross-validation folds, *sed* seed, *res* results.

Lemmatizer and the LATINO *Lemma Tagger LemmaGen*.

Every stemmer/lemmatizer was applied to the document corpus, as shown in Figure 4.8. In every iteration (over the list of stemmers/lemmatizers) of the for loop the input stemmer/lemmatizer is applied to the preprocessed ADC data object using the *Stem/Lemma Tagger Hub* by generating new features with name “Stem” on every elementary token. In order to use the stemmed values together with Part-of-Speech tags we constructed (for every token) new features named “Stem with PoS” using the *Add Computed Token Features* widget. These features were later used in the *CrossValidation* subprocess to generate the BoW models. The values of the “Stem with PoS” features were constructed using the *Add Computed Token Features* widget as a combination of stems/lemmas and PoS tags: “Stem_PoS tag”.

What follows is the *CrossValidation* subprocess which was executed for every cross-validation fold. In this subprocess, a sparse BoW model dataset and the *BowModelConstructor* instance were generated by the *Construct BoW Dataset and BoW Model Constructor*. We constructed unigrams (N-grams where N is equal to 1) out of “Stem with PoS” features from word tokens, while disregarding stop words in the process. The Bag-of-Words vector space was calculated using the TF-IDF (Term Frequency - Inverse Document Frequency) weighting scheme.

The test sparse matrix is required to be constructed using the same parameter settings, same IDFs and same vocabulary as the training sparse matrix. Therefore, we applied the BoW constructor object (output of the *Construct BoW Dataset and BoW Model Constructor*) to the test ADC data object using the *Create BoW Dataset using the BoW Model Constructor*.

After calculating the training and testing sparse matrices, we fitted the maximum entropy classifier to the training dataset using the *Train Classifier Hub*. Next, the *Apply Classifier Hub* received the trained classifier object and returned predicted class probabilities for every new document from the input test dataset. The *Extract Actual and Predicted Values* widget used these probabilities and constructed pairs of actual and predicted classes. These pairs were returned from the *CrossValidation* subprocess and used for calculating different metrics, as shown in Figure 4.8.

Table 4.2 shows the results of the presented stemming/lemmatization evaluation workflow. The first row in the table shows the classification results without applying a stemmer/lemmatizer. We see that the usage of a stemmer/lemmatizer increases the performance of the classifier. The best results were obtained using the NLTK's *WordNet Lemmatizer*, which achieved a slightly higher classification accuracy on the Kenyan elections dataset compared to other applied stemmers and lemmatizers.

Table 4.2: Stemming/lemmatization evaluation on the Kenyan elections dataset.

Library	Stemmer/Lemmatizer	F1 score	Class. accuracy	AUC
/	no stemmer	0.94	94.16%	0.94
NLTK	RSLP Stemmer	0.96	95.67%	0.96
NLTK	Snowball Stemmer	0.96	96.10%	0.96
NLTK	ISRI Stemmer	0.96	95.67%	0.96
NLTK	WordNet Lemmatizer	0.96	96.32%	0.96
NLTK	Lancaster Stemmer	0.95	95.02%	0.95
NLTK	Porter Stemmer	0.95	94.59%	0.95
Latino	Stem Tagger Snowball	0.95	94.81%	0.95
Latino	Lemma Tagger LemmaGen	0.95	95.24%	0.95

4.4 Part-of-Speech Tagger Comparison for Text Categorization Workflow

In this section we describe a workflow for the evaluation of different Part-of-Speech taggers on the Kenyan elections dataset. We start the processing with the *Document acquisition* and *Document preprocessing* steps, described in Section 4.1. In order to perform a comparison of Part-of-Speech taggers we exceptionally omit the PoS tagger from the document preprocessing step and apply it later in the cross-validation subprocess.

In the experiments we compared five different PoS taggers: *English Maximum Entropy PoS Tagger* (from LATINO library) and *PoS Affix Tagger, PoS N-gram Tagger, PoS Brill's rule-based Tagger, PoS Classifier-based Tagger* (from NLTK library). As shown in Figure 4.9, PoS tagging widgets from NLTK require tagged sentence data as input, which is used for training the PoS tagger. The TextFlows platform already contains several tagged datasets, which come as a part of the NLTK library, and can be added to workflows through the *NLTK Document Corpus* widget. In this use case we used the annotated Brown corpus[1].

The training process involves inspecting the tag of each word and storing the most likely tag for any word in a dictionary, which is stored inside the tagger. As it happens, once we have processed several thousand words of English text, most new words will be nouns. In cases when the NLTK PoS tagger is unable to assign a tag from its learned lookup table, it can use a backoff tagger from its input. As shown in Figure 4.9 we have set the *PoS Default Tagger* as the backoff tagger for all NLTK PoS taggers. The *PoS Default Tagger* assigns the input tag (e.g., "NN", which is the PoS tag representing a noun) to every single word. Whenever the initial PoS tagger cannot assign a tag to a token it will invoke the input backoff tagger and thus tag the token as a noun. This improves the robustness of the language processing system. The *PoS Classifier-based Tagger* widget also requires input of a classifier, which is learned to predict PoS tags based on the pre-annotated dataset. Every PoS tagger was applied to the document corpus, as shown in Figure 4.10. In every iteration

[1]Description of the Brown corpus is available at `http://www.nltk.org/book/ch02.html#tab-brown-sources`

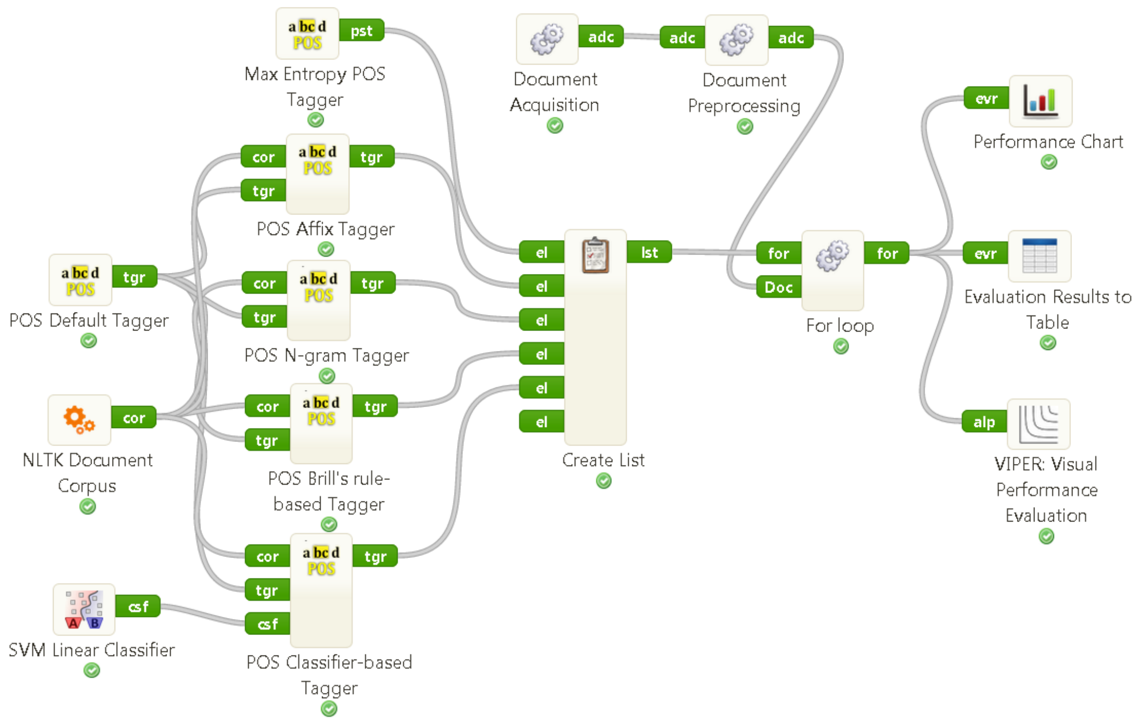


Figure 4.9: The workflow used for evaluation of 5 different PoS taggers from various text mining libraries on a text categorization problem. The workflow is publicly available at `http://textflows.perovsek.com/workflow/37/`.

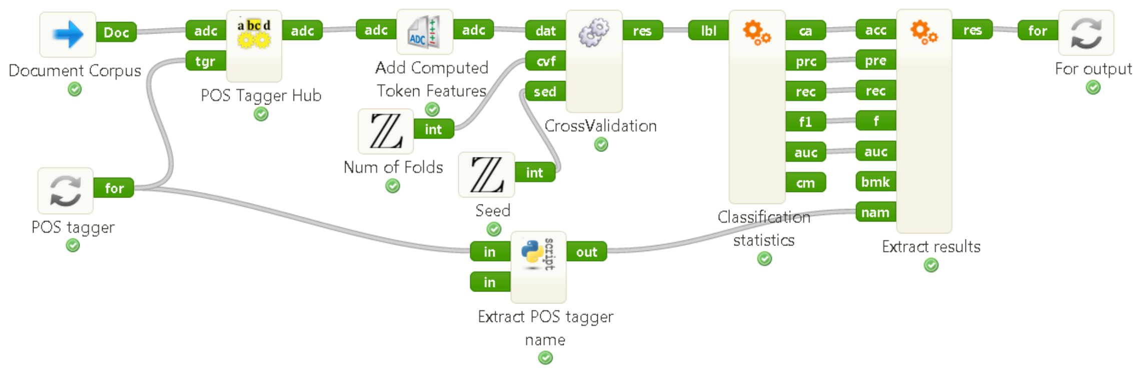


Figure 4.10: The subprocess workflow representing the methodology in every fold for the used 10-fold cross-validation. First, the input PoS tagger is applied to the document corpus. Next, the construction of the training and testing sparse matrices is performed. Then, a linear SVM classifier is fitted to the training dataset and used to predict class probabilities for every new document from the input test dataset. Last, the subprocess returns constructed pairs of actual and predicted classes.

(over the list of PoS taggers) of the for loop the input PoS tagger is applied to the tokenized sentences of preprocessed ADC data object using the *PoS Tagger Hub* by generating new features with name “PoS tag” on every elementary (word) token. In order to use PoS tags together with stemmed values, we constructed (for every token) new features named “Stem with PoS” using the *Add Computed Token Features* widget. These features were later used in the CrossValidation subprocess to generate the BoW models. The values of the “Stem with PoS” features were constructed using the *Add Computed Token Features* widget as a combination of stems and PoS tags: “Stem_PoS tag”.

Next, 10-fold stratified cross-validation was performed on the generated PoS tagged ADC data objects. Similarly as in the classifier evaluation use case, all cross-validations were performed using the same seed in order to ensure the data was equally split for all PoS taggers. The methodology behind the *CrossValidation* subprocess, which is executed on every cross-validation fold, is similar to the methodology presented in Figure 4.4. The only two differences are that cross-validation does not receive a classifier on its input—instead it always uses scikit-learn’s linear SVM classifier—and that the *Construct BoW Dataset and BoW Model Constructor* widget uses features constructed by the *Add Computed Token Features* widget instead of stemmed values.

Table 4.3 shows the results of the presented PoS tagger evaluation workflow. The first row in the table shows the classification results without applying a PoS tagger (see row 3 of Table 4.1). We see that the usage of a PoS tagger increases the performance of the classifier. The best results were obtained using the NLTK’s *PoS Classifier Based Tagger*, which in combination with the LATINO’s *Maximum Entropy Classifier* achieved a slightly higher classification accuracy on the Kenyan elections dataset compared to the other PoS taggers.

The experiments were run using the same resources as in the classifier evaluation example—a virtual machine with a setting of one core and 4GB of RAM. The execution time of the entire workflow was 1,913 seconds, where 158 seconds were used for document acquisition and preprocessing, while the for loop which compares PoS taggers took 1,747 seconds to execute.

Table 4.3: Part-of-Speech tagger evaluation on the Kenyan elections dataset.

Library	Tagger	Recall	Precision	F1 score	Classification accuracy	AUC
/	no PoS tagger	0.98	0.93	0.95	95.24%	0.95
LATINO	Maximum Entropy PoS Tagger	0.98	0.94	0.96	96.10%	0.96
NLTK	PoS Affix Tagger	0.98	0.94	0.96	95.67%	0.96
NLTK	PoS Ngram Tagger	0.98	0.95	0.96	96.10%	0.96
NLTK	PoS Brill Tagger	0.97	0.93	0.95	95.24%	0.95
NLTK+ scikit-learn	PoS Classifier Based Tagger (using SVM Linear Classifier)	0.98	0.95	0.96	96.32%	0.96

4.5 Outlier Document Detection in Categorized Document Corpora Workflow

In this section we propose a workflow for detecting atypical, unusual and/or irregular documents on the Kenyan elections dataset. The idea behind irregularity detection in categorized document corpora is based on early noise filtering approaches by [102], who used a classifier as a tool for detecting noisy instances in data. Noise detection approaches identify irregularities and errors in data and are therefore suitable also for detecting atypical documents in categorized document corpora, which can be considered as outliers of their own document category.

The aim of the NoiseRank (ensemble-based noise detection and ranking) methodology, proposed by [95], [103], is to support domain experts in identifying noisy, outlier or erroneous data instances. The user should be able to select the noise detection algorithms to be used in the ensemble-based noise detection process. We have implemented this methodology as a workflow in TextFlows, which now offers widgets implementing classification and saturation noise filters, and enables the inclusion of external user specific noise detection algorithms available as web services. Figure 4.11 presents the NoiseRank workflow using the implemented classifiers used for class noise detection.

The NoiseRank methodology workflow returns a visual representation of a list of potential outlier instances, ranked according to the decreasing number of noise detection algorithms which identified an instance as noisy, due to its classification into a class different from its own class label. The ability of NoiseRank to obtain atypical documents was tested on the Kenyan elections corpus. The implemented voting-based irregularity detection method uses four different classifiers acting as noise detection algorithms by identifying misclassified instances.

As in the experiments of Section 4.2 and Section 4.4 we ran the workflow on a virtual machine with 1 CPU core and 4GB of RAM. The execution time of the entire workflow was 202 seconds, where 148 seconds were used for document acquisition and preprocessing, while the BoW model construction and NoiseRank widget took 52 seconds to execute.

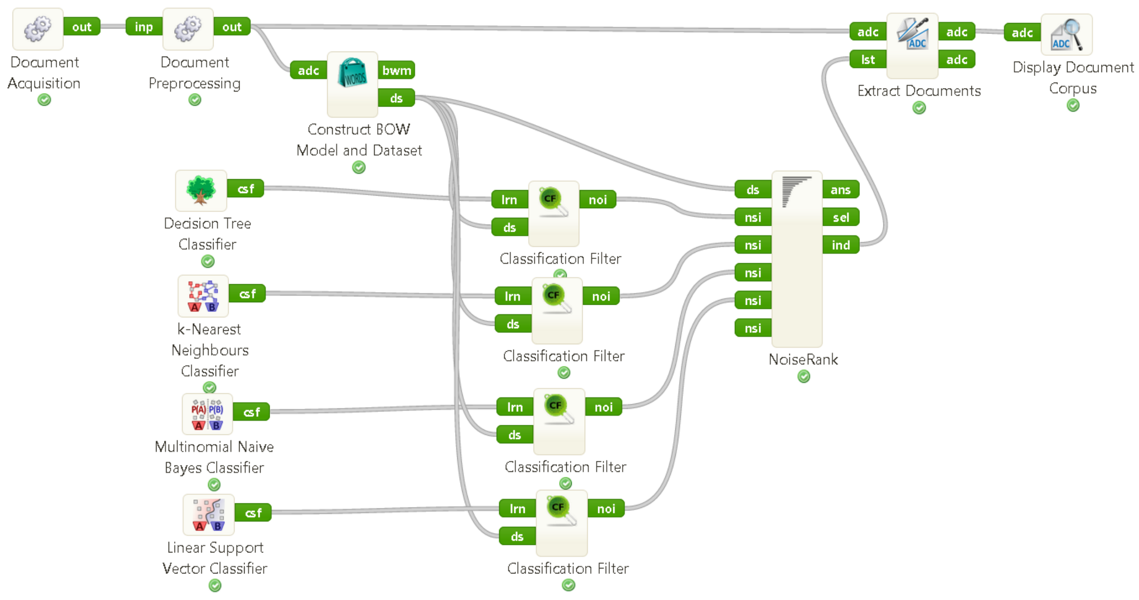
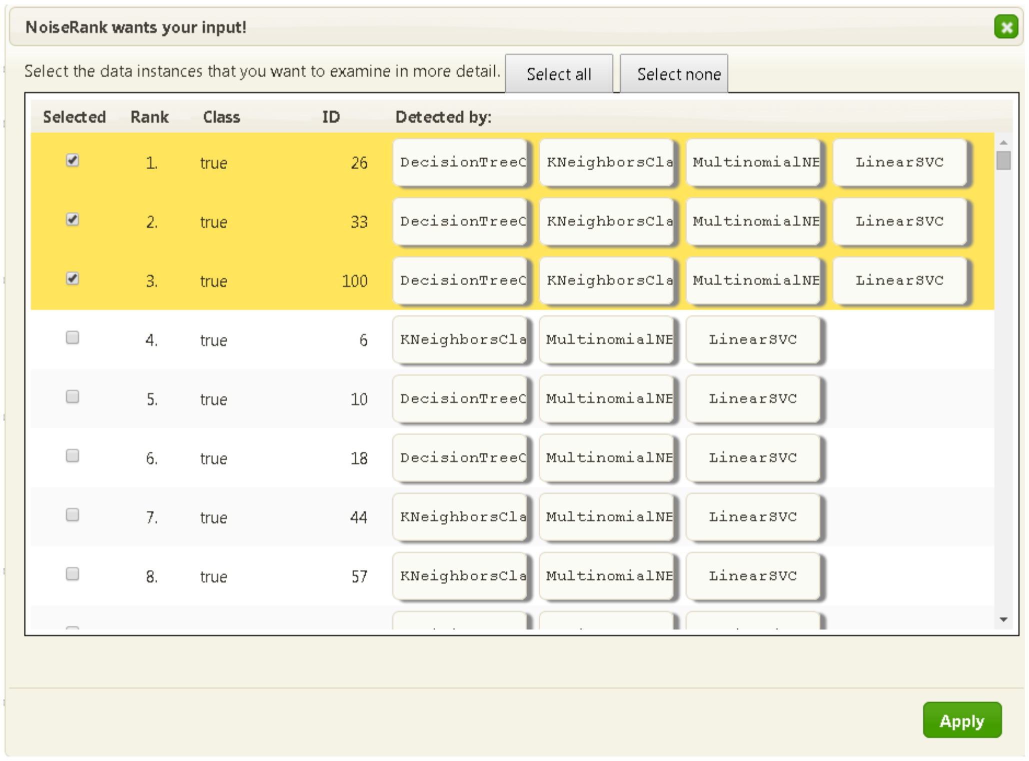


Figure 4.11: Example workflow of the NoiseRank methodology, which is available at `http://textflows.perovsek.com/workflow/29/`.

Figure 4.12 shows the obtained set of atypical/irregular articles grouped and ranked according to the number of noise detection algorithms that identified them as irregular.


NoiseRank wants your input!
Select the data instances that you want to examine in more detail.
Select all
Select none
Apply

Selected	Rank	Class	ID	Detected by:			
✓	1.	true	26	DecisionTreeC	KNeighborsCla	MultinomialNE	LinearSVC
✓	2.	true	33	DecisionTreeC	KNeighborsCla	MultinomialNE	LinearSVC
✓	3.	true	100	DecisionTreeC	KNeighborsCla	MultinomialNE	LinearSVC
☐	4.	true	6	KNeighborsCla	MultinomialNE	LinearSVC	
☐	5.	true	10	DecisionTreeC	MultinomialNE	LinearSVC	
☐	6.	true	18	DecisionTreeC	MultinomialNE	LinearSVC	
☐	7.	true	44	KNeighborsCla	MultinomialNE	LinearSVC	
☐	8.	true	57	KNeighborsCla	MultinomialNE	LinearSVC	

Figure 4.12: The NoiseRank interactive widget where the user gets a visual representation of a list of top-ranked potentially noisy instances, which are misclassified according to a decreasing number of elementary noise detection algorithms which identified the instance as noisy. The user can decide which documents he wishes to exclude from the document corpus.

Chapter 5

Cross-Domain Literature Mining Scenario

Given its immense growth, scientific literature can be explored to reveal new discoveries, based on yet uncovered relations between knowledge from different, relatively isolated fields of research specialization. This chapter proposes a bisociation-based text mining approach, which demonstrates that cross-domain literature mining can be effective for cross-domain knowledge discovery and, thus, helps the experts in searching for hidden links that connect seemingly unrelated domains. The proposed cross-domain literature mining functionality, including text acquisition, text preprocessing, and bisociative cross-domain literature mining facilities, is made publicly available within the web-based platform TextFlows, which supports visual construction and execution of text mining and natural language processing (NLP) workflows. To this end, TextFlows was connected to the human-computer interface of system CrossBee [17]–[19]. In the methodology presented in this chapter, the CrossBee web application—which was originally developed as an off-the-shelf solution for finding bisociations bridging two domains—is used as a user interface to facilitate bridging term discovery through sophisticated document visualization and exploration.

The chapter is organized as follows. Section 5.1 illustrates the problem of bridging term ranking. Since this part of the thesis continues the work in the area of literature mining proposed by Juršič et al. [19], [70], we provide a brief description of CrossBee [17], [19]—an off-the-shelf solution for finding bisociations bridging two user defined domains/literatures—in Section 5.2. We also provide an overview of the underlying methodology, as well as describe the elementary and ensemble heuristics used in bridging term discovery. In Section 5.3 we show how the complex CrossBee methodology was implemented as a workflow in TextFlows. In this scenario, the platform acts as the enabling technology for implementing the developed cross-domain link discovery approach. Moreover, we study a new type of elementary heuristics for B-term ranking in Section 5.4. The newly proposed approach uses banded matrices [104] to discover structures that reveal the relations between the rows (representing documents) and columns (representing words/terms) of a given data matrix (representing a set of documents). We evaluate the newly proposed heuristics on two medical benchmark problems. Finally, in Section 5.5, we propose an extension of the CrossBee methodology, which incorporates background knowledge—controlled vocabularies—into the bridging term discovery process that, as a side effect, also narrows down the list of potential bridging terms. We also provide an evaluation of the developed extended methodology.

5.1 Problem Definition and Motivation

Understanding complex phenomena and solving difficult problems often requires knowledge from different domains to be combined and cross-domain associations to be taken into account. These kinds of context crossing associations, called *bisociations* [13], are often needed for creative, innovative discoveries. Bisociative knowledge discovery is a challenging task motivated by a trend of over-specialization in research and development, which usually results in deep—but relatively isolated—knowledge islands. Scientific literature too often remains closed and cited only in professional sub-communities. In addition, the information that is related across different contexts is difficult to identify using associative approaches, like the standard association rule learning [105] known from the data mining and machine learning literature. Therefore, the ability of literature mining methods and software tools to support the experts in their knowledge discovery processes—especially in searching for yet unexplored connections between different domains—is becoming increasingly important. Cross-domain literature mining is closely related to bisociative knowledge discovery as defined by [106]. Assuming two different domains of interest, a crucial step in cross-domain knowledge discovery is the identification of interesting bridging terms (B-terms), appearing in both literatures, which carry the potential of revealing the links connecting the two domains.

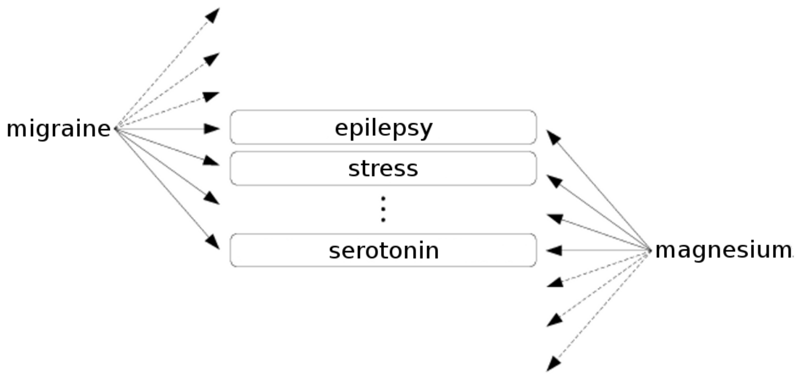


Figure 5.1: Bridging term discovery when exploring migraine and magnesium document corpora, with B-terms as identified by [63] in the middle.

In cross-domain knowledge discovery, estimating which of the terms have a high potential for interesting discoveries is a challenging research question. It is especially important for cross-context scientific discovery such as understanding complex medical phenomena or finding new drugs for not yet fully understood illnesses. In our approach we focus on the closed discovery process, where two disjointed domains A and C are specified at the beginning of the discovery process and the main goal is to find bridging terms (see Figure 5.1) which support validation of the novel hypothesized connection between the two domains.

Given this motivation, the main functionality of the approaches, presented in the following sections, is bridging term (B-term) discovery implemented through ensemble based term ranking, where an ensemble heuristic composed of several elementary heuristics is constructed for term evaluation.

5.2 CrossBee Approach to Cross-domain Literature Mining

This section briefly describes previous work, proposed by Juršič et al., in bisociative knowledge discovery for the task of literature mining. The CrossBee tool [17]–[19] is an off-the-shelf solution for finding bisociations bridging two user defined domains/literatures. CrossBee is a system that not only suggests bridging terms using the ensemble ranking methodology but also helps the experts when searching for hidden links that connect two seemingly unrelated domains. In addition to this core functionality, supplementary functionalities and content presentations were added, which make the CrossBee web application a user-friendly tool for ranking and exploration of prospective cross-context links. This enables the user not only to spot but also to efficiently investigate discovered cross-domain links.

As proposed by Juršič et al., CrossBee uses an ensemble based ranking approach for bridging term (B-term) detection. The main concept of this methodology is illustrated in Figure 5.2, where term ranking is performed using an ensemble heuristic. Figure 5.3 shows the list of B-terms ranked by ensemble voting of heuristics, where the presented ranked list is the actual output produced by the CrossBee bridging term exploration system using the gold standard dataset in literature mining, i.e. the combined migraine-magnesium dataset [63]. The ranked list of B-term candidates, shown in Figure 5.3, provides the user some additional information including the ensemble's individual base heuristics votes and term's domain occurrence statistics in both domains.

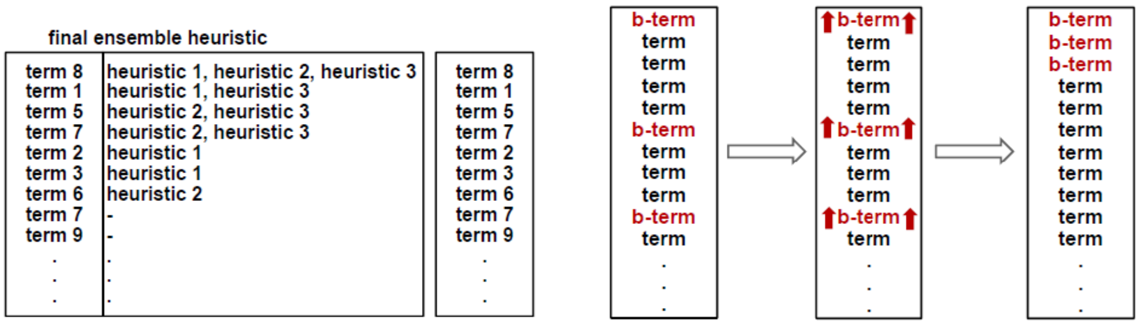


Figure 5.2: Term ranking approach: first, ensemble heuristics vote for terms, next, terms are sorted according to their potential B-term (as shown on left). Consequently, bridging terms with the highest bridging term potential should receive the highest scores (as shown on the right side). The figure is taken from [18] and [19].

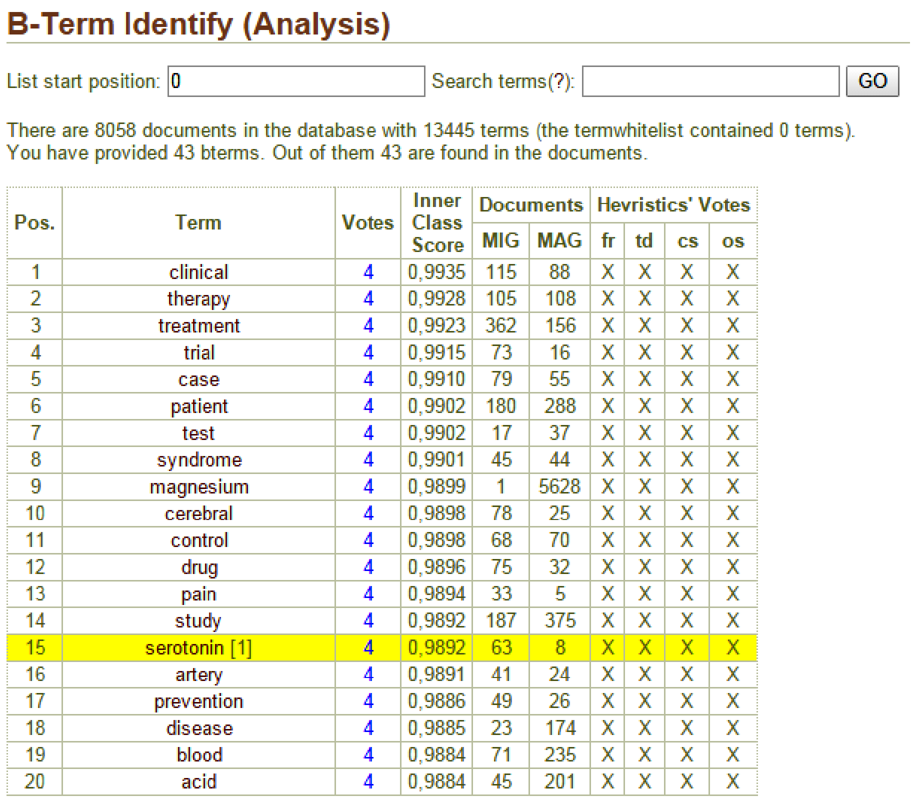
5.2.1 Methodology Outline

This section shortly describes the top-level procedural steps of the CrossBee methodology for bridging term (B-term) discovery. As shown in Figure 5.4 the entire methodology—implemented through ensemble-based term ranking, where an ensemble heuristic composed of six elementary heuristics was used—can be divided into seven individual steps, which are described below:

1. The goal of the document acquisition is to acquire documents of the two domains, label them with domain labels and prepare the text for the next step of the methodology.

2. The document preprocessing step is responsible for applying standard text preprocessing to the document corpus. The main parts are tokenization, stopword tagging, and token stemming/lemmatization.

3. The heuristic specification step enables detailed specification of the heuristics to be used for B-term ranking. The user specifies one or more heuristics, which are to be applied to evaluate the B-term candidates.



B-Term Identify (Analysis)

List start position: 0 Search terms(?): GO

There are 8058 documents in the database with 13445 terms (the termwhitelist contained 0 terms).
You have provided 43 bterms. Out of them 43 are found in the documents.

Pos.	Term	Votes	Inner Class Score	Documents		Hevristics' Votes			
				MIG	MAG	fr	td	cs	os
1	clinical	4	0,9935	115	88	X	X	X	X
2	therapy	4	0,9928	105	108	X	X	X	X
3	treatment	4	0,9923	362	156	X	X	X	X
4	trial	4	0,9915	73	16	X	X	X	X
5	case	4	0,9910	79	55	X	X	X	X
6	patient	4	0,9902	180	288	X	X	X	X
7	test	4	0,9902	17	37	X	X	X	X
8	syndrome	4	0,9901	45	44	X	X	X	X
9	magnesium	4	0,9899	1	5628	X	X	X	X
10	cerebral	4	0,9898	78	25	X	X	X	X
11	control	4	0,9898	68	70	X	X	X	X
12	drug	4	0,9896	75	32	X	X	X	X
13	pain	4	0,9894	33	5	X	X	X	X
14	study	4	0,9892	187	375	X	X	X	X
15	serotonin [1]	4	0,9892	63	8	X	X	X	X
16	artery	4	0,9891	41	24	X	X	X	X
17	prevention	4	0,9886	49	26	X	X	X	X
18	disease	4	0,9885	23	174	X	X	X	X
19	blood	4	0,9884	71	235	X	X	X	X
20	acid	4	0,9884	45	201	X	X	X	X

Figure 5.3: The ensemble heuristic based ranking page indicating by a cross (X) which elementary heuristics have identified the term as potential B-term. This example shows the top 20 ranked terms from the migraine-magnesium domain according to the selected heuristics.

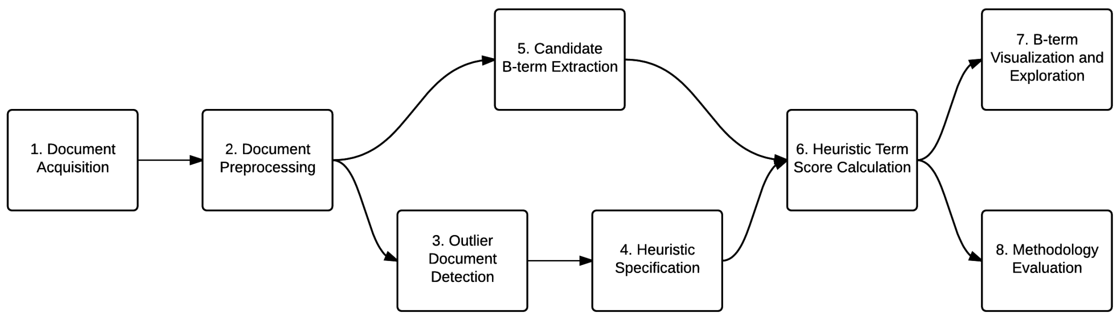

1. Document Acquisition
2. Document Preprocessing
5. Candidate B-term Extraction
3. Outlier Document Detection
4. Heuristic Specification
6. Heuristic Term Score Calculation
7. B-term Visualization and Exploration
8. Methodology Evaluation

Figure 5.4: Methodological steps of the CrossBee methodology as proposed by Juršič et al. [17], [19], [70].

4. The candidate B-term extraction step takes care of extracting the terms which are later scored by the specified heuristics. There are various parameters which control which kind of terms are extracted from the documents (e.g., the maximal number of tokens to be joined together as a term, minimal term corpus frequency, and similar).
5. Heuristic term score calculation is the most important step of the methodology. It takes the list of extracted B-term candidates and the list of specified heuristics and calculates a heuristic score for each candidate term for each heuristic.
6. Visualization and exploration is the final step of the methodology. It has three main functionalities.
7. Methodology evaluation was introduced as an additional step, which can be used during the development of the methodology. Its purpose is to calculate and visualize various metrics that were used to assess the quality of the methodology.

Evaluation of the methodology was actually performed on two problems: the standard migraine-magnesium problem well-known in LBD, and a more recent autism-calcineurin literature mining problem. As we use these datasets also in our experiments we will describe them in more detail in Section 5.2.4. In the following section we will describe the elementary and ensemble heuristics, proposed by Juršič et al. [19], [70], which define the heuristic specification step, in more detail.

5.2.2 Heuristics for Bridging Term Discovery

Different elementary and ensemble heuristics used for B-term ranking are available in CrossBee. The heuristics are defined as functions that numerically evaluate the term quality by assigning it bisociation score to a term (measuring the potential that a term is actually a B-term). For the definition of an appropriate set of heuristics, we define a set of special (mainly statistical) properties of terms, which aim at distinguishing B-terms from regular terms; thus, these heuristics can also be viewed as advanced term statistics.

Formally, a heuristic is a function with two inputs, i.e. a set of domain labeled documents D and a term t appearing in these documents, and one output, i.e. a score that represents the term's bisociation potential. All the heuristics operate on the data retrieved from the documents in text preprocessing. While term ranking using the scores calculated by an ideal heuristic should result in ranking all the B-terms at the top of the ranked list, this ideal scenario is not realistic; nevertheless, ranking by heuristic scores should at least increase the proportion of B-terms at the top of the ranked term list.

In the work of Juršič et al. [19], [70] base heuristics are divided into four sets: six frequency based heuristics, four TF-IDF based, three similarity based, and eight outlier based heuristics. Most of the developed heuristics work fundamentally in a similar way—they all manipulate solely the data present in term and document vectors and derive the terms bisociation score. The exceptions to this are the outlier-based heuristics, which first evaluate outlier documents and only later use the information from the term vectors for B-term evaluation. Using these base heuristics they developed the ensemble heuristic described below.

5.2.3 Ensemble Heuristic

The ensemble heuristic for bridging term discovery, which was constructed based on the experiments [19], [70], is constructed as a sum of two parts: the ensemble voting score s_t^{vote} and the ensemble position score s_t^{pos}, which are summed together to give the final ensemble score for every term in the corpus vocabulary. Each term score represents the term's potential for linking the two disjointed domains.

The ensemble voting score (s_t^{vote}) of a given term t is an integer, which denotes how many base heuristics voted for the term: each term can get a score $s_{t_j}^{vote} \in \{0, 1, 2, ..., k\}$, where k is the number of base heuristics used in the ensemble. The ensemble voting score of term t_j that is at position p_j in the ranked list of n terms is computed as a sum of individual heuristics' voting scores:

$$s_{t_j}^{vote} = \sum_{i=1}^{k} s_{t_j,h_i}^{vote} = \sum_{i=1}^{k} \begin{cases} 1, & p_j < n/3 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

The ensemble position score (s_t^{pos}) is calculated as an average of position scores of individual base heuristics. For each heuristic h_i, the term's position score s_{t_j,h_i}^{pos} is calculated as $\frac{n-p_j}{n}$, which results in position scores being in the interval $[0, 1)$. For an ensemble of k heuristics, the ensemble position score is computed as an average of individual heuristics' position scores:

$$s_{t_j}^{pos} = \frac{1}{k} \sum_{i=1}^{k} s_{t_j,h_i}^{pos} = \frac{1}{k} \sum_{i=1}^{k} \frac{n - p_j}{n} \tag{5.2}$$

The final ensemble score is computed as:

$$s_t = s_t^{vote} + s_t^{pos} \tag{5.3}$$

The definitions[1] of final set of elementary heuristics they included in the ensemble are given below, while a detailed justification for the choice of this particular combination of heuristics is presented in [19].

- $outFreqRelRF(t) = \frac{countTerm_{D_{RF}}(t)}{countTerm_{D_u}(t)}$: relative frequency of term t in the outlier document set detected by the Random Forest classifier,
- $outFreqRelSVM(t) = \frac{countTerm_{D_{SVM}}(t)}{countTerm_{D_u}(t)}$: relative frequency of term t in outlier document set detected by the Support Vector Machine classifier,
- $outFreqRelCS(t) = \frac{countTerm_{D_{CS}}(t)}{countTerm_{D_u}(t)}$: relative frequency of term t in the outlier document set detected by the Centroid Similarity classifier,
- $outFreqSum(t) = countTerm_{D_{CS}}(t) + countTerm_{D_{RF}}(t) + countTerm_{D_{SVM}}(t)$: sum of frequencies of term t in all three outlier document sets,
- $tfidfDomnSum(t) = tfidf_{D_1}(t) + tfidf_{D_2}(t)$: sum of term TF-IDF weights of term t in the two domains,
- $freqRatio(t) = \frac{countTerm_{D_u}(t)}{countDoc_{D_u}(t)}$: term to document frequency ratio.

[1]Due to a large number of heuristics and auxiliary functions, we use the so-called camel casing multi-word naming scheme for easier distinction; names are formed by word concatenation and capitalization of all non-first words (e.g., *freqProdRel* and *tfidfProduct*).

5.2.4 Experimental Setting

Juršič et al. [19], [70] performed evaluation based on two datasets (or two domain pairs, since each dataset consists of two domains)—the migraine-magnesium dataset [63] and the autism-calcineurin [69] dataset—which was viewed as a training and test dataset, respectively. The training dataset was the dataset they employed when developing the methodology, i.e. for creating a set of base heuristics as well as for creating the ensemble heuristic. The results of the evaluation on the training dataset were interpreted carefully due to a danger of overfitting the dataset, as described in [19], [70]. The test dataset was used for the evaluation of the methodology in a real-life setting.

Table 5.1: B-terms for the migraine-magnesium dataset identified by Swanson [63].

1 5 ht	16 convulsive	31 prostaglandin
2 5 hydroxytryptamine	17 coronary spasm	32 prostaglandin e1
3 5 hydroxytryptamine receptor	18 cortical spread depression	33 prostaglandin synthesis
4 anti aggregation	19 diltiazem	34 reactivity
5 anti inflammatory	20 epilepsy	35 seizure
6 anticonvulsant	21 epileptic	36 serotonin
7 antimigraine	22 epileptiform	37 spasm
8 arterial spasm	23 hypoxia	38 spread
9 brain serotonin	24 indomethacin	39 spread depression
10 calcium antagonist	25 inflammatory	40 stress
11 calcium blocker	26 nifedipine	41 substance p
12 calcium channel	27 paroxysmal	42 vasospasm
13 calcium channel blocker	28 platelet aggregation	43 verapamil
14 cerebral vasospasm	29 platelet function	
15 convulsion	30 prostacyclin	

The well-researched migraine-magnesium domain pair [63] was used as a training set. In the literature-based discovery process Swanson managed to find more than 60 pairs of articles connecting the migraine domain with the magnesium deficiency via 43 bridging concepts (B-terms), which are listed in Table 5.1.[2] In testing the developed methodology Juršič et al. aimed at rediscovering the 43 B-terms by ranking them as high as possible in the ranked list of potential B-terms that include Swanson's B-terms and terms that are not in the Swanson's B-term list.

Table 5.2: B-terms for the autism-calcineurin dataset identified by Petrič et al. [69].

1 synaptic	6 bcl 2	11 22q11 2
2 synaptic plasticity	7 type 1 diabetes	12 maternal hypothyroxinemia
3 calmodulin	8 ulcerative colitis	13 bombesin
4 radiation	9 asbestos	
5 working memory	10 deletion syndrome	

For the test dataset Juršič et al. used the autism-calcineurin domain pair, proposed in [69]. Like Swanson, Petrič et al. also provide B-terms, 13 in total (listed in Table 5.2), whose importance in connecting autism to calcineurin (a protein phosphatase) is discussed and confirmed by the domain expert. In view of searching for B-terms, this dataset has a relatively different dimensionality compared to the migraine-magnesium dataset. On the one hand it has only about one fourth of the B-terms defined, while on the other

[2]Note that Swanson did not state that this was an exclusive list, hence there may exist other important bridging terms which he did not list.

hand, it contains more than 40 times as many potential B-term candidates. Therefore, the ratio between the actual B-terms and the candidate terms is substantially lower—approximately by factor 160, i.e. the chance to find a B-term among the candidate terms if picking it at random is 160 times lower in the autism-calcineurin dataset than in the magnesium-migraine dataset. Consequently, finding the actual B-terms in the autism-calcineurin dataset is much more difficult compared to the migraine-magnesium dataset.

Both datasets, retrieved from the PubMed database using a keyword query, were formed of titles or abstracts of scientific papers returned by the query. However, they used an additional filtering condition for selecting the migraine-magnesium dataset. For fair comparison with Swanson's experiments they selected only the articles published before the year 1988 as this was the year when Swanson published his research about this dataset and consequently making an explicit connection between the migraine and magnesium domains.

Table 5.3: Comparison of some statistical properties of the two datasets used in the experiments, as presented in [19], [70].

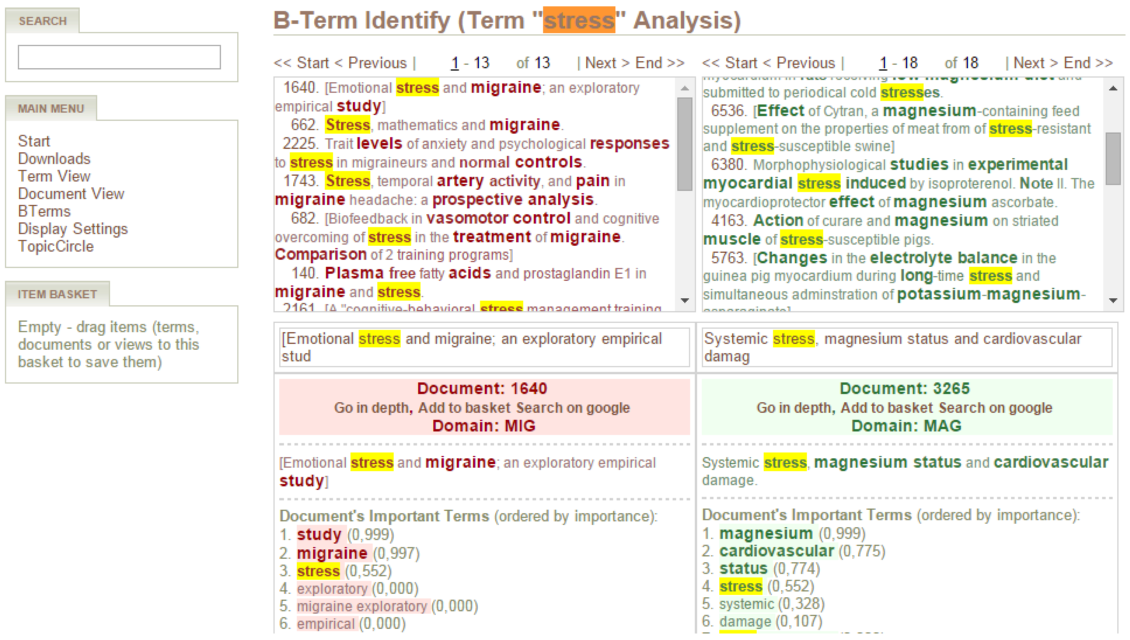
		migraine-magnesium	autism-calcineurin
Retrieval	Source	PubMed	PubMed
	Query terms	"migraine"-"magnesium"	"autism"-"calcineurin"
	Additional conditions	Year < 1988	/
	Part of paper used	Title	Abstract
Document Statistics	Number	8,058 (2,415-5,633)	15,243 (9,365-5,878)
	Doc. with B-term	394 (4.89%)	1,672 (10.97%)
	Avg. words per doc.	11	180
Term statistic	Avg. term per doc.	7	173
	Distinct terms	13,525	322,252
	B-term candidates	1,847	78,805
	Defined B-terms	43	13

Table 5.3 states some properties for comparing the two datasets that were used in the evaluation. One of the major differences between the datasets is the length of an average document since only the titles were used in the migraine-magnesium dataset, while the full abstracts were used in the autism-calcineurin case. Consequently, also the number of distinct terms and B-term candidates is much larger in the case of the autism-calcineurin dataset.

5.2.5 The CrossBee HCI Interface

The CrossBee website is built on top of the CrossBee library [17], [70]. From this perspective CrossBee is firstly, a functional enhancement and secondly, a wrapping of this functionality into a practical web user interface especially designed for the requirements of bisociation discovery. After the ensemble heuristic computation, the user is presented with a ranked list of B-term candidates as seen in Figure 5.3. The list provides the user some additional information including the ensemble's individual base heuristics votes and term's domain occurrence statistics in both domains.

The user-friendly CrossBee web interface can be used to efficiently investigate cross-domain links ranked by the ensemble-based ranking methodology. CrossBee's document focused exploration empowers the user to filter and order the documents by various criteria, including detailed document view that provides a more detailed presentation of a single document including various term statistics. Methodology performance analysis supports the evaluation of the methodology by providing various data which can be used to mea-



SEARCH

MAIN MENU

Start
Downloads
Term View
Document View
BTerms
Display Settings
TopicCircle

ITEM BASKET

Empty - drag items (terms, documents or views to this basket to save them)

B-Term Identify (Term "stress" Analysis)

<< Start < Previous | 1 - 13 of 13 | Next > End >>

1640. [Emotional stress and migraine; an exploratory empirical study]
662. Stress, mathematics and migraine.
2225. Trait levels of anxiety and psychological responses to stress in migraineurs and normal controls.
1743. Stress, temporal artery activity, and pain in migraine headache: a prospective analysis.
682. [Biofeedback in vasomotor control and cognitive overcoming of stress in the treatment of migraine. Comparison of 2 training programs]
140. Plasma free fatty acids and prostaglandin E1 in migraine and stress.

<< Start < Previous | 1 - 18 of 18 | Next > End >>

submitted to periodical cold stresses.
6536. [Effect of Cytran, a magnesium-containing feed supplement on the properties of meat from of stress-resistant and stress-susceptible swine]
6380. Morphophysiological studies in experimental myocardial stress induced by isoproterenol. Note II. The myocardioprotector effect of magnesium ascorbate.
4163. Action of curare and magnesium on striated muscle of stress-susceptible pigs.
5763. [Changes in the electrolyte balance in the guinea pig myocardium during long-time stress and simultaneous adminstration of potassium-magnesium-

[Emotional stress and migraine; an exploratory empirical stud

Systemic stress, magnesium status and cardiovascular damag

Document: 1640
Go in depth, Add to basket Search on google
Domain: MIG

[Emotional stress and migraine; an exploratory empirical study]

Document's Important Terms (ordered by importance):
1. study (0,999)
2. migraine (0,997)
3. stress (0,552)
4. exploratory (0,000)
5. migraine exploratory (0,000)
6. empirical (0,000)

Document: 3265
Go in depth, Add to basket Search on google
Domain: MAG

Systemic stress, magnesium status and cardiovascular damage.

Document's Important Terms (ordered by importance):
1. magnesium (0,999)
2. cardiovascular (0,775)
3. status (0,774)
4. stress (0,552)
5. systemic (0,328)
6. damage (0,107)

Figure 5.5: One of the useful features of the CrossBee interface is the side-by-side view of documents from the two domains under investigation. The analysis of the term "stress" from the migraine-magnesium domain is shown. The presented view enables efficient comparison of two documents, the left one from the migraine domain and the right one from the magnesium domain.

sure the quality of the results, e.g., data for plotting the ROC curves. High-ranked term emphasis marks the terms according to their bisociation score calculated by the ensemble heuristic. When using this feature all high-ranked terms are emphasized throughout the whole application thus making them easier to spot (see different font sizes in Figure 5.5). B-term emphasis marks the terms defined as B-terms by the user (yellow terms in Figure 5.5). Domain separation is a simple but effective option which colors all the documents from the same domain with the same color, making an obvious distinction between the documents from the two domains (different colors in Figure 5.5). User interface customization enables the user to decrease or increase the intensity of the following features: high-ranked term emphasis, B-term emphasis and domain separation.

The user can inspect the actual appearances of the selected term in both domains, using the side-by-side document inspection as shown in Figure 5.5. In this way, they can verify whether his rationale behind selecting this term as a bridging term can be justified based on the contents of the inspected documents.

Note that the CrossBee web interface was designed for end users who are not computer scientists or data miners and who prefer using the system by following a fixed sequence of predefined methodological steps. However, for a more sophisticated user, the weakness of CrossBee is the lack of possibility to experiment with different settings as well as the lack of possibility to extend the methodology with new ideas and then compare or evaluate the developed approaches. As another weakness, the CrossBee web application does not offer a downloadable library and documentation distribution or extensive help. These weaknesses were among the incentives for our new developments, resulting in the TextFlows implementation and its elaborate mechanisms for detecting and exploring bisociative links between the selected domains of interest.

5.3 The CrossBee Methodology as a Workflow in TextFlows

In this section we show how the complex CrossBee methodology, presented in Section 5.2, was implemented as a workflow in the TextFlows platform. To ensure the best user experience in the process of bridging term discovery we have combined the visual programming interface of the TextFlows platform with the bridging term exploration system CrossBee. In such setting TextFlows is responsible for workflow construction and execution, while CrossBee provides only the user interface (described in Section 5.2.5) for term and document visualization, which supports the expert in finding relevant documents and exploration of the top-ranked bisociative terms.

The user starts the bridging term discovery process in TextFlows by either constructing a new workflow for cross-domain discovery or by opening an existing workflow. In the former case, the user is required to input either a PubMed query or a file with documents from the two domains, where each line contains a document with exactly three tab-separated entries: (a) document identifier, (b) domain acronym, and (c) the document text. The user is able to tailor the preprocessing steps to his own needs by simply altering the workflow using the TextFlows visual programming user interface, which enables simple addition, connection and removal of components from the workflow canvas. In this way, the user can also modify the ensemble of elementary heuristics, outlier documents identified by external outlier detection software, the already known bisociative terms (B-terms), and others, which was previously not possible in the CrossBee tool. When the user runs the workflows (by clicking the run button) the system starts with the process of text preprocessing, followed by the computation of elementary heuristics, the ensemble bisociation scores and term ranking.

After performing the calculation of bisociative potentials for every term in the vocabulary in TextFlows, the user is directed to the user-friendly tool CrossBee where one can efficiently investigate cross-domain links pointed out by the ensemble-based ranking methodology. CrossBee's document focused exploration empowers the user to filter and order the documents by various criteria, including detailed document view that provides a more detailed presentation of a single document including various term statistics. Methodology performance analysis supports the evaluation of the methodology by providing various data which can be used to measure the quality of the results, e.g., data for plotting the ROC curves. High-ranked term emphasis marks the terms according to their bisociation score calculated by the ensemble heuristic. When using this feature all high-ranked terms are emphasized throughout the whole application thus making them easier to spot (see different font sizes in Figure 5.5). B-term emphasis marks the terms defined as B-terms by the user (yellow terms in Figure 5.5). Domain separation is a simple but effective option which colors all the documents from the same domain with the same color, making an obvious distinction between the documents from the two domains (different colors in Figure 5.5). User interface customization enables the user to decrease or increase the intensity of the following features: high-ranked term emphasis, B-term emphasis and domain separation.

As described in Section 5.2.1 the top-level overview of the CrossBee methodology consists of the following steps: document acquisition, document preprocessing, heuristics specification, candidate B-term extraction, heuristic terms scores calculation, and visualization and exploration. Figure 5.6 shows the described pipeline now implemented as an actual executable workflow in the online cloud-based workflow composition and execution environment TextFlows. The workflow for cross-domain literature mining is publicly available[3] for sharing and reuse within the TextFlows platform. The workflow integrates the specification and computation of heuristics, described in Section 5.2.2, and is connected to

[3]`http://textflows.perovsek.com/workflow/7/`

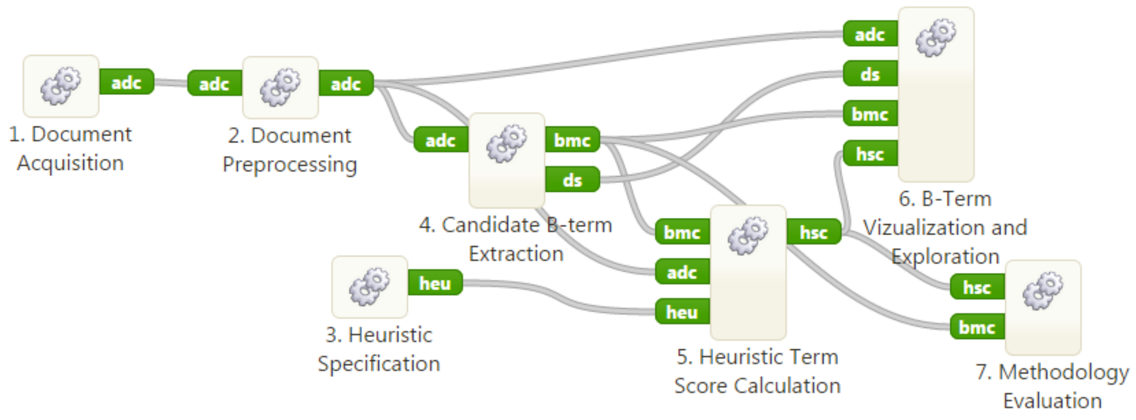

1. Document Acquisition
adc
adc
2. Document Preprocessing
adc
adc
4. Candidate B-term Extraction
bmc
ds
3. Heuristic Specification
heu
bmc
adc
heu
5. Heuristic Term Score Calculation
hsc
adc
ds
bmc
hsc
6. B-Term Vizualization and Exploration
hsc
bmc
7. Methodology Evaluation

Figure 5.6: Methodological steps of the CrossBee cross-domain literature mining approach as a workflow in the TextFlows platform. This workflow is publicly available at `http://textflows.perovsek.com/workflow/7/`.

the term exploration interface of the online system CrossBee, which supports the user in advanced document exploration by facilitating document analysis and visualization. Since the proposal of such workflow was already presented in [19] and our main contribution is the adapted implementation in the TextFlows platform, we will not include the description of individual steps of the workflow in our main part of the thesis. Instead, we will describe the entire underlying pipeline of natural language processing (NLP) and literature-based discovery (LBD) components (from the workflow, shown in Figure 5.6) in Appendix A.1.

5.4 Banded Matrix Based Heuristics

In this section we study a new type of heuristics for B-term ranking, which use banded matrices [104] to discover structures which reveal the relations between the rows (representing documents) and columns (representing words/terms) of a given data matrix (representing a set of documents). We use this information in developing new heuristics for evaluating words/terms according to their bridging term (B-term) potential. In addition, the method enables the identification of document outliers, but this is out of the scope of this thesis.

The methodology works by first encoding the documents from the two domains into the standard Bag-of-Words (BoW) vector representation and then transforming the binary matrix of BoW vectors to its banded structure. The proposed banded matrices methodology is based on the assumption that similar documents, as well as the words that appear in the same document, will appear closer to each other in the matrix and will therefore form 'clusters' along the main diagonal of the matrix in its banded form[4]. Our work is based on the intuition that terms that connect different domains will be positioned at the edges of clusters from different domains: we have developed different heuristics that should be able to identify these B-terms by ranking them high in the ranked list of terms with high potential for cross-context link discovery. We introduce below the banded matrices, and follow this with a toy example illustrating the approach.

5.4.1 Definition of Banded Matrices

Uncovering structures that reveal the nature of relations between rows and columns of data matrices is an important step towards solving real-world problems, as binary data occur

[4]A correspondence between bi-clustering and banded structures has been shown in [107].

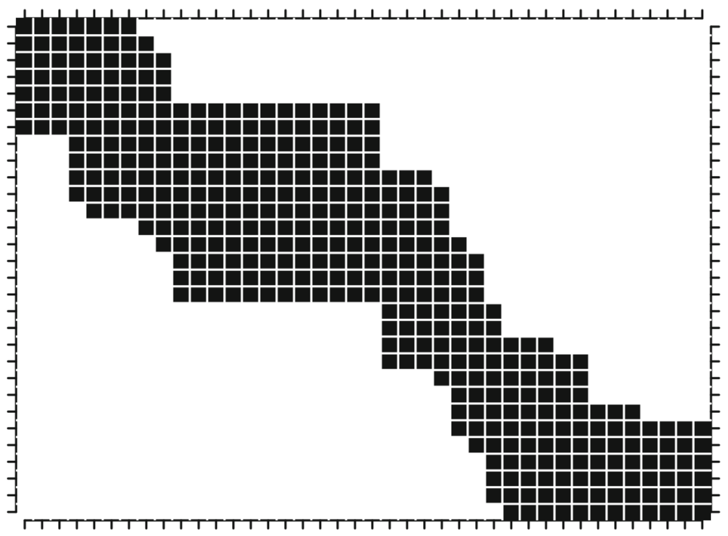


Figure 5.7: An example of a fully banded matrix. The figure is taken from [104].

in numerous real-world applications. Recent research in social networks, bioinformatics, and human genomics has shown the benefits of banded representations of matrices [107]. These representations have contributed to bringing huge performance boosts in various mathematical operations, including matrix multiplication.

To explain the algorithm that transforms a matrix into its banded structure we first need to define the basic concepts. A binary matrix has a banded structure if we can find a permutation of rows and columns such that the ones (1s) exhibit a staircase pattern down the rows along the leading diagonal, as illustrated in Figure 5.7.

Definition 5.1. *A binary matrix M is fully banded if there exists a permutation of rows κ and a permutation of columns π such that:*

1. *for every row i in M_κ^π the entries with 1s occur in consecutive column indices $a_i, a_i + 1, \ldots, b_i$, and*
2. *the values of starting indices for 1s in successive rows (i and i+1) satisfy $a_i \leq a_{i+1}$ and $b_i \leq b_{i+1}$.*

A necessary precondition for (1) to hold is that matrix M satisfies the *consecutive-ones property*: a binary matrix satisfies this property if it is possible to order the columns so that in every row the non-zero entries occur in consecutive positions.

As banded structured matrices cannot be expected to arise in noisy real-world environments, we need to redefine the problem in the sense that it is applicable to a wider range of real-world situations. We aim to minimize the number of transformations one needs to perform on a binary matrix to unveil a banded structure. The number of such transformations will measure how far the matrix is from being fully banded. The algorithm presented in the next section (following the motivating example) aims to solve this optimization problem.

5.4.2 A Motivating Example

Let us have two sets of documents A and C. For the purpose of explaining the methodology we constructed a small ideal-world dataset, which consists of 6 clusters of documents, 3 of which belong to domain A, while the others belong to domain C. Initially, we took a set of 120 different randomly selected words and randomly divided them into 6 clusters, so that there were no intersections, i.e. each word belonged to one cluster only. The number of possible words per cluster was 20, while each document in the cluster was randomly assigned only 15 of these words. Using a banded matrix algorithm presented in the next section this document set would first be transformed into a fully banded matrix form shown in Figure 5.9 and finally (by applying the discovered row and column permutations on the initial matrix) into the structure shown in Figure 5.10.

In order to illustrate our methodology, we randomly chose 8 words from each of the two domains A and C to act as artificially defined, preselected bridging terms. This effect was achieved by inserting the preselected terms into every document in every cluster with a 10% chance, thus spoiling the original clean separation of words within documents of different clusters. The resulting matrix showing documents as rows, and words as columns, is depicted in Figure 5.8, where the green vertical lines represent the artificially inserted B-terms. As the aim of our method is to identify the bridging terms, we conducted experiments to check how the designed preselected terms will be ranked by our heuristics.

Having used our methodology on the ideal-world toy domain of Figure 5.8, we got the result shown in Figure 5.10. The green vertical lines represent the terms which were deliberately acting as bridging terms in this experiment. As can be seen from Figure 5.10, similar documents (documents from the same cluster) and similar terms (terms that are contained in the same document cluster) are located close to each other. As a result, the 'clusters' along the matrix leading diagonal are clearly visible. Note that the preselected bridging terms occur mainly on the transitions between the clusters. All of our heuristics (explained in the next section) correctly assigned a B-term score greater than 0 only to the preselected bridging terms, which served as a proof-of-concept for the toy experiment.

Let us now consider a single document only. A document from domain A (represented

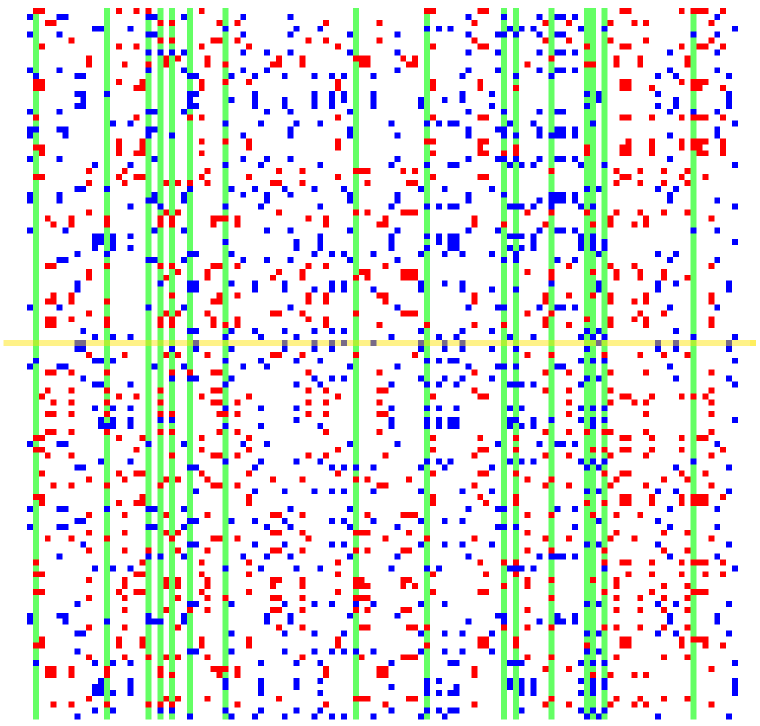


Figure 5.8: Documents (rows) and words (columns) in an ideal-world domain. The color of a row indicates the domain of a document (blue for domain A and red for domain C).

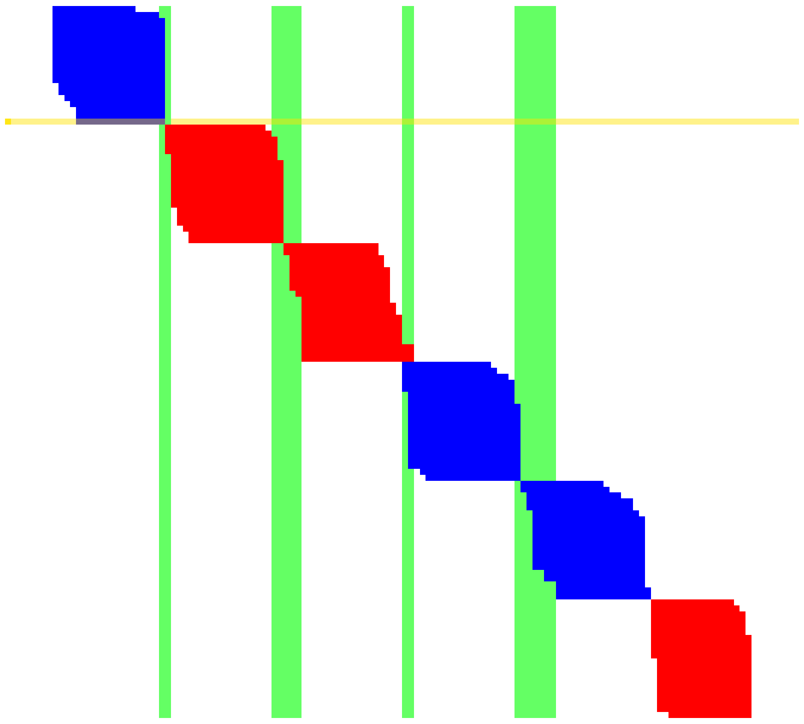


Figure 5.9: Matrix of documents shown after the transformation of documents in Figure 5.8 into a fully banded matrix structure. Rows represent the documents, while columns represent the terms. The green vertical lines represent the terms which were inserted as potential bridging terms to the documents.

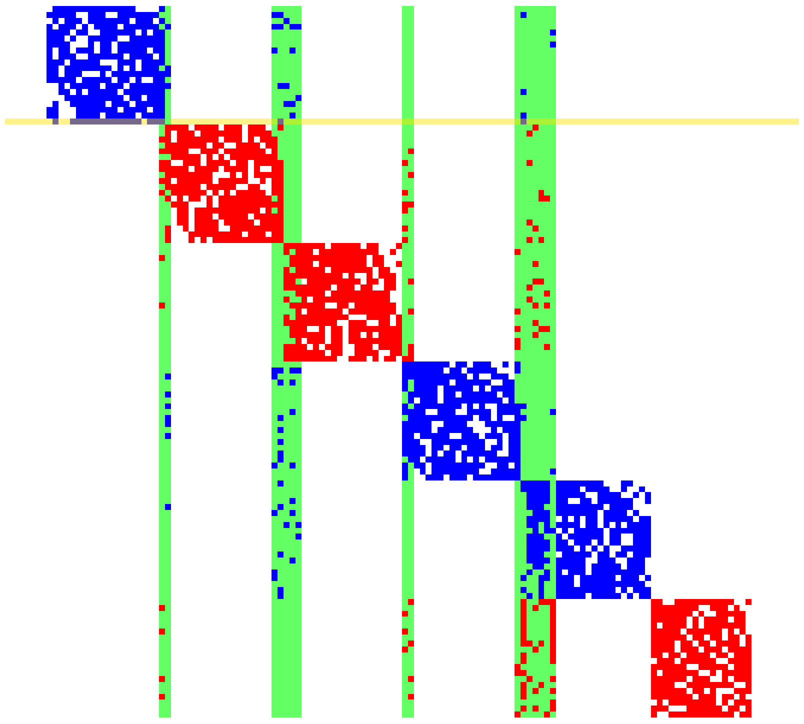


Figure 5.10: Final result of our methodology: matrix of documents from Figure 5.8 permuted using row and column permutations obtained from the transformation of the matrix into its fully banded structure.

with a horizontal yellow line on Figures 5.8, 5.9 and 5.10 consist of the following words: *magnesium blood cell prophylaxi relationship lithium red calcium effect sodium control membrane measurement potential perfuse* ***ophthalmoplegic simultaneous***, where the first 15 words were randomly selected from the document's cluster term set, while the two words in bold were randomly inserted from the preselected set of bridging terms. The blue dots on the horizontal yellow line in Figure 5.10 consequently symbolize the above words. According to the banded structure of the matrix (see Figure 5.9) the words ***simultaneous*** and ***ophthalmoplegic*** belong to word clusters of domains A and C, respectively. While the observed document belongs to domain A, the term ***ophthalmoplegic*** is representative of the documents from domain C. Therefore, our methodology should be able to identify this term as a potential B-term. In contrast, as the word ***simultaneous*** is used in the documents from the same domain A, it should not be considered as a B-term. Indeed, our heuristics (presented in the next section) have only identified ***ophthalmoplegic*** as a B-term. Figure 5.9 shows the final result of the banded matrix algorithm and is used in the next section for the explanation of our heuristics.

5.4.3 A Methodology for B-term Ranking Using Banded Matrices

Our approach is designed to find links between two domains, named A and C, by exploring the bridging terms that connect these two separate domains. The methodology works as follows. First, we preprocess the documents from the two domains using standard text mining techniques [1]. This is performed through a number of steps: stop-word removal, stemming or lemmatization, usage of synonym dictionaries, construction of n-grams of words and, finally, transformation to a Bag-of-Words representation.

Next, the result of the preprocessing step, i.e. the binary matrix of "Bag-of-Words" vectors (the BoW matrix), is transformed to its banded matrix structure. Finally, we sort the terms according to their scores representing their bridging term potential, computed according to the new heuristics described below. In the following subsections, each step of the proposed bridging term detection and ranking methodology is described in detail.

5.4.4 Constructing a Banded Structure Using a Bidirectional Minimum Banded Augmentation Algorithm

The optimization problem addressed to make a banded matrix is labeled Bidirectional Minimum Banded Augmentation (MBA) [104] and is defined as follows:

Problem 5.1. *Given a binary matrix M, find the minimum number of bidirectional flips (flips from both 1s to 0s and 0s to 1s) so that M becomes fully banded.*

Algorithm 5.1: Bidirectional Minimum Banded Augmentation (MBA) algorithm

```
1. Find fixed permutation of columns $\pi$.
2. Solve the consecutive-ones property on the column permuted matrix $M^\pi$.
3. Resolve Sperner conflicts (defined later in this section) between rows in $M^\pi$.
4. Sort the rows in $M^\pi$ and return the row permutation.
```

The presented MBA algorithm discovers a single band by first fixing the column permutations of the data matrix before proceeding with the rest of the algorithm. A good permutation of columns tends to put similar columns (i.e. terms) closer to each other. We use the Jaccard coefficient as a column similarity measure: $J(M^a, M^b) = \frac{M^a \cap M^b}{M^a \cup M^b}$. In our example, this similarity measure returns the highest value of 1 when two terms occur in the same set of documents. We used the spectral ordering algorithm [108] to find the fixed column permutation π of matrix M.

Next, the algorithm deals with solving the consecutive-ones properties on rows of matrix M^π, which is an essential step in finding the row permutation κ. Solving the consecutive-ones problem for row M_i^π with bidirectional flips corresponds to solving the maximum sub-array problem on matrix W_i^j [104], defined as follows:

$$W_i^j := \begin{cases} 1 & \text{if } M_i^j = 1 \\ -1 & \text{if } M_i^j = 0 \end{cases} .$$

The objective of solving the maximum sub-array problem is to find the sub-array of the matrix that has the maximum sum of numbers. This problem can be solved in linear time with respect to the size of the array using the scan-line algorithm [109]. This method returns the interval boundaries which we use to solve the consecutive-ones problem in M_i^π by setting the fields in M_i^π between the boundaries to 1 and others to 0.

Next, the algorithm deals with removing the Sperner conflicts between the rows of matrix M^π. A matrix has Sperner conflicts if its rows do not form a Sperner family of intervals:

Lemma 5.2. *Two rows $M_i = [a, b]$ and $M_j = [a', b']$ with consecutive-ones property, where $i \neq j$, form a Sperner family of intervals if they are overlapping such that $(a \geq a' \vee b' \geq b) \wedge (a' \geq a \vee b \geq b')$.*

Additional flips on rows of M^π need to be made in order to ensure that rows have the Sperner family of intervals property.

Lemma 5.3. *Let $\hat{M}$ be the binary matrix M augmented with $M_{ij} = M_i \backslash M_j$ for every two rows $M_i \subset M_j$. Note that M is fully banded if and only if $\hat{M}$ has the consecutive-ones property (proof in [104]).*

To eliminate all Sperner conflicts between row intervals of M^π, the algorithm has to go through all extra rows described in $\hat{M}$ and make sure that they have the consecutive-ones property. This can be done by solving the maximum sub-array problem on the extra rows of $\hat{M}$. We perform additional flips in order for the rows to obtain consecutive-ones property. Lastly, we update the rows in M^π according to the changes made over $\hat{M}$ in order to get a banded matrix.

Finally, the algorithm sorts rows $[a, b]$ of M^π in an ascending order of as, while deciding ties with the ascending order of their bs. The result of the algorithm is a banded matrix M_{band}, along with details of the row and column permutations that were performed. We use these permutations on our original matrix M, as the objective is to produce a matrix without distorting the data (i.e. without making the bidirectional flips). In the next section, we present the heuristics for calculating the B-terms potential scores.

5.4.5 New Heuristics for B-term Potential Evaluation

Here we describe the details of the three banded matrix-based heuristics, which we propose for computing the bridging term potential scores. After completing the step of term score computation, we sort the terms according to the values of one of the heuristics and present the top-ranked terms (hopefully representing the most interesting B-term candidates) to the expert. The designed heuristics should favor B-terms over non-B-terms by pushing interesting B-term candidates higher to the top of the ranked term list. For easier definition of the proposed heuristics we define variable d_{idx} to represent the row index of document d in the banded matrix M_{band} and t_{idx} to represent the column index of term t in M_{band}. Note that in order to compute the score of the proposed heuristics, we distinctively take into account the document-term matrix in two forms, banded (as shown in Figure 5.9) and full (as shown in Figure 5.10).

- Heuristic 1: This is a frequency based heuristic for computing the B-term potential. If all document-term pairs in the t_{idx}-th column of matrix M_{Band}, which equal to 1, belong to the same domain, we denote this domain as D_1. Note that in such a case, the t_{idx}-th column, which represents term t in the banded matrix, should be "single-colored" in the matrix visualization in Figure 5.9. If the documents in the M_{Band} for term t do not belong to the same domain, the heuristic returns score 0 for this term ($h1score(t) := 0$). Otherwise, the score of Heuristic 1 for term t is defined as:

  $$h1score(t) := countDoc_{D_2}(t),$$

  where $countDoc_{D_2}(t)$ is the number of documents that contain term t and belong to domain D_2 (do not belong to domain D_1) in the matrix shown in Figure 5.10. This heuristic is based on the assumption that terms which strongly represent one domain (the single-colored column in the banded matrix of Figure 5.9), and at the same time there are many documents from the other domain that contain these terms, have a higher chance of being the bridging terms between the two domains.

- Heuristic 2: This is also a frequency based heuristic. Similarly as described in Heuristic 1, if all documents for which the t_{idx}-th column of matrix M_{Band} equals to 1 belong to the same domain, we label this domain as D_1. Otherwise, the heuristic returns score 0 for term t ($h2score(t) := 0$). Heuristic 2 score for term t is defined as:

  $$h2score(t) = \frac{countDoc_{D_2}(t)}{countOnDiagDoc_{D_1}(t)},$$

  where $countDoc_{D_2}(t)$ is the number of documents from domain D_2 that contain term t, and $countOnDiagDoc_{D_1}(t)$ is the count of document-term pairs equaling 1 in the t_{idx}-th column of the banded matrix M_{Band}: $countOnDiagDoc_{D_1}(t) := |\{d_{idx}; d \in D_1 \wedge M_{Band}(d_{idx}, t_{idx}) = 1\}|$. Therefore, for term t $h2score(t)$ is the ratio between the count of documents that belong to domain D_2 and the documents in the M_{Band} for term t which belong to domain D_1. The intuition behind this heuristic is that a term that strongly represents a given domain according to the banded matrix, and is at the same time also contained in many documents of the other domain, should also have a high B-term potential score.

- Heuristic 3: If the documents in M_{Band} for term t do not belong to the same domain, this heuristic returns a score of 0 ($h3score := 0$). Otherwise, we label this domain as D_1 and define the Heuristic 3 score for term t as follows:

$$h3score(t) = \frac{countOnDiagDoc_{D_1}(t)}{countDoc_{D_1}(t)} * countDoc_{D_2}(t),$$

where $countOnDiagDoc_{D_1}(t)$ is the count of document-term pairs in the t_{idx}-th column of banded matrix M_{Band}, and where documents belong to domain D_1: $countOnDiagDoc_{D_1}(t) = |\{d_{idx}; d \in D_1 \wedge M_{Band}(d_{idx}, t_{idx}) = 1\}|$; $countDoc_{D_1}(t)$ denotes the number of documents from domain D_1 that contain term t, while $countDoc_{D_2}(t)$ is the number of documents from domain D_2 that contain term t. The bridging term potential score for term t is the ratio of documents from domain D_1 that are contained in the 'diagonal cluster' multiplied by the number of documents from the other domain. The intuition behind this heuristic is that for term t, the more the term represents a domain (has a large proportion of document on the diagonal of the banded matrix) and also the more documents from the other domain that contain t exist, the higher the potential of term t to be a bridging term between the two domains.

5.4.6 Experiments and Results

This section presents the evaluation of the newly presented heuristics for literature-based discovery. We have applied different base and ensemble heuristics on two problems, which were described in Section 5.2.4: the standard migraine-magnesium literature mining benchmark problem used in the Swanson's experiments [63], and a more recent example of using literature mining for uncovering the nature of relations that might contribute to better understanding of autism, originated by [68] and [69].

Evaluation Procedure

The key aspect of the evaluation is the assessment of how well the proposed ensemble heuristic performs when ranking the terms. Two evaluation measures were used in the evaluation of the developed methodology: the standard Area under the Receiver Operating Characteristic analysis and the number of B-terms found among the first 5, 10, 20, 100, 500 and 2,000 terms in the heuristics' ranked list of terms.

First, we compared the heuristics using the Area under the Receiver Operating Characteristic (AUROC) analysis [110]. The Receiver Operating Characteristic (ROC) space is defined by two axes, where the horizontal axis scales from zero to the number of non-B-terms, and the vertical axis from zero to the number of B-terms. An individual Receiver Operating Characteristic (ROC) curve, representing a single heuristic, is constructed in the following way:

- Sort all the terms by their descending heuristic score.
- For every term of the term list do the following: if a term is a B-term, then draw one vertical line segment (up) in the ROC space, or else draw one horizontal line segment (right) on the ROC space.
- If a heuristic outputs the same score for many terms, we cannot sort them uniquely. In such case, we draw a line from the current point p to the point $p+(nb, b)$, where nb is the number of non-B-terms and b is the number of terms that are B-terms among the terms with the same bisociation score. In this way we may produce slanted lines, if such an equal scoring term set contains both B-terms and non-B-terms.

AUROC is defined as the percentage of the area under ROC curve, i.e. the area under the curve divided by the area of the whole ROC space. Besides AUROC we also list the interval of AUROC which tells how much each heuristic varies among the best and the worst sorting of a possibly existing equal scoring term set. This occurs due to the fact that some heuristics do not produce unambiguous ranking of all the terms. Several heuristics assign the same score to a set of terms—including both the actual B-terms as well as non B-terms—which results in the fact that unique sorting is not possible.[5] In the case of equal scoring term sets, the inner sorting is random (which indeed produces different performance estimates), however the constructed ROC curve corresponds to the average ROC curve over all such possible random inner sortings.

From the expert's point of view, the ROC curves and AUROC statistics are not the most crucial information about the quality of a given heuristic. While in general it still holds that a higher AUROC reflects a better heuristic, we are more interested in the ranking from the perspective of the domain expert (the end-user of our system) who is usually more interested in questions like: (a) how many B-terms are likely to be found among the first n terms in a ranked list (where n is a selected number of terms the expert is willing to inspect, e.g., 5, 20 or 100), or (b) how much one can trust a heuristic if a new dataset is explored. Therefore, we also performed an evaluation using an alternative user oriented approach, which evaluates the ranking results adapted to the user's needs. This evaluation estimates how many B-terms can be found among the first 5, 10, 20, 100, 500 and 2,000 terms on the ranked list of terms produced by a heuristic.

Results on the Migraine-magnesium Dataset

Table 5.4 shows the comparison of ranking performance for all the base banded matrix-based heuristics, as well as all heuristics proposed by Juršič [19], [70], on the migraine-magnesium dataset. The heuristics are ordered by their AUROC. The second and third column in the table represent heuristics' average AUROC score[6] and its AUROC interval, respectively.

As mentioned, such AUROC evaluation does not necessarily align well with the methodology evaluation from a user's perspective. Therefore, the right side of Table 5.4 shows the results of an alternative user-oriented evaluation approach, which shows how many B-terms were found among the first 5, 10, 20, 50, 100, 200, 500, 1,000 and 2,000 terms on the ranked list of terms produced by a heuristic.

The three banded matrix-based heuristics, described in Section 5.4.5, resulted in very favorable results on the training migraine-magnesium domain (as seen in Table 5.4). Heuristic 3, which we consider the most complete of the three, outperforms all except the outFreqRelRf heuristic (according to the AUROC score) and is the heuristic which finds the most B-terms among the top 200 ranked terms. The other two banded matrix-based heuristics are also among the best ranked heuristics. However, e.g., if the expert limits himself to inspect only the first 20 terms, he will find 0 B-terms in the ranked term list using any of the newly presented heuristics. Consequently, from the user's perspective, the new heuristics do not outperform the others.

[5]In such cases, the AUROC calculation can either maximize the AUROC by sorting all the B-terms in front of all the other terms inside equal scoring sets or minimize it by putting the B-terms at the back.

[6]In contrast to the results reported by [17], [19], [70], the AUROC scores presented in this section take into account only the terms which appear in both domains. This results in lower AUROC scores, which are thus not directly comparable between the studies. The reason for this approach is in the definition of a bridging term, where the term is required to appear in both domains, as it cannot form a connection otherwise.

Table 5.4: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the migraine-magnesium dataset.

Heuristic Name	AUROC		Number of B-terms among the top n ranked terms								
	Average	Interval	5	10	20	50	100	200	500	1,000	2,000
outFreqRelRf	60.32%	1.39%	0.24	0.48	0.95	1	3.72	7.92	15.8	30.33	43
heuristic3	**59.63%**	**2.22%**	**0**	**0**	**0**	**1**	**3**	**9**	**14.92**	**28.48**	**43**
outFreqRelSum	59.44%	0.62%	0	0.36	1	1.64	3.54	7	15	27	43
outFreqRelSvm	58.91%	1.25%	0.12	0.24	0.48	1	1.63	5.92	14.69	29	43
heuristic2	**58.36%**	**4.81%**	**0**	**0**	**0**	**1**	**6**	**7.91**	**12.65**	**29.62**	**43**
outFreqRf	57.50%	10.94%	0	0	0	0	0	3.91	16.03	27.86	43
freqDomnRatioMin	57.34%	4.71%	0.14	0.28	0.57	1.42	2.83	5.66	14	28	43
heuristic1	**56.35%**	**8.73%**	**0**	**0**	**0**	**1**	**2**	**4.48**	**16.5**	**26.58**	**43**
outFreqSum	56.17%	3.97%	0	0	0	0	0	2	14.28	27.11	43
outFreqSvm	55.25%	9.34%	0	0	0	0	0.32	3	14.09	26.14	43
outFreqRelCs	54.29%	1.50%	0	0	1	1	2.69	5.07	11	27	43
freqDomnProdRel	53.23%	3.08%	0	0	0	0	0	6	14	27	43
outFreqCs	52.34%	10.51%	0	0	0	0	0	1.43	15.62	24.67	43
tfidfDomnSum	52.11%	2.69%	0	0	0	0	1	2	11	26.14	43
tfidfAvg	51.31%	3.63%	0	0	1	1.79	3.11	5.75	11.84	20.9	43
freqDomnProd	51.20%	3.36%	0	0	0	0	1	3	13.17	27.16	43
tfidfDomnProd	51.18%	2.69%	0	0	0	0	1	3	13.5	27	43
freqRatio	50.51%	39.26%	0	0	1	1	4	5	11.65	23.09	43
appearInAllDomains	50.00%	50.00%	0.11	0.23	0.46	1.15	2.3	4.6	11.49	22.98	43
tfidfSum	49.65%	3.63%	0	0	0	0	0	1	9	25.36	43
freqTerm	49.60%	3.78%	0	0	0	0	0	1	8.91	25.49	43
freqDoc	49.55%	3.82%	0	0	0	0	0	1	8.03	24.79	43

Results on the Autism-calcineurin Dataset

In this section we show how our newly proposed heuristics based on banded matrices performs on the autism-calcineurin domain. As discussed, the dimensionality of the autism-calcineurin dataset is considerably different and less favorable compared to the migraine-magnesium dataset.

Table 5.5 shows that the performance of the proposed base heuristics based on banded matrices significantly changes compared to the migraine magnesium dataset (Table 5.4). These heuristics are now among the average ranked (Heuristic 2 even among the worst ranked) according to the AUROC evaluation. This could be due to the fact that the dimensionality of the autism-calcineurin dataset is considerably less favorable compared to the migraine-magnesium dataset. As the BoW space in this domain is much sparser, the banded matrix of this space contains many more ‘clusters’ along the leading diagonal, which are also smaller. Consequently, the heuristics are presented with a more difficult job of identifying terms that connect different domains, which should be positioned at the edges of clusters in the banded matrix.

The right side of the Table 5.5 shows the results of an alternative user oriented evaluation approach—the number of B-terms among the first n terms on the ranked list. The results show that all heuristics (except two) fail to present the user with a single B-term among the top ranked 50 terms, making the discovery of new connections between the two domains an extremely difficult task. As described in Section 5.2.3, Juršič et al. [19], [70] proposed an approach using ensembles, where different heuristics vote for terms, to overcome this issue. They show that a carefully chosen ensemble of heuristics improves the quality of the best ranked terms. In the next section we propose an alternative approach, which uses background knowledge to improve the ranking capabilities of the base heuristics, and thus improves the quality of the best ranked terms of the ensemble as well.

Table 5.5: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the autism-calcineurin dataset.

Heuristic Name	AUROC		Number of B-terms among the top n ranked terms										
	Average	Interval	5	10	20	50	100	200	500	1,000	2,000	5,000	all
freqRatio	95.10%	0.16%	1	1	1	1	1	1	1	3	5	8.99	13
tfidfSum	88.78%	0.05%	0	0	0	0	1	1	1	2	4	5	13
tfidfDomnProd	88.61%	0.05%	0	0	0	0	0	0	1	1	4	6	13
tfidfDomnSum	88.33%	0.02%	0	0	0	0	1	1	2	2	4	5	13
freqTerm	87.80%	0.80%	0	0	0	0	1	1	1	2	3	5	13
freqDomnProd	87.69%	0.73%	0	0	0	0	0	0	0	1	2	6	13
freqDomnProdRel	85.77%	0.69%	0	0	0	0	0	0	0	1	1	6	13
outFreqSum	80.86%	7.94%	0	0	0	0	0	1	1	3	5	7	13
outFreqCs	80.50%	10.05%	0	0	0	0	0	1	1	1	4	7.17	13
freqDoc	79.01%	2.53%	0	0	0	0	0	1	1	2	2	5	13
outFreqRf	78.11%	12.55%	0	0	0	0	0	1	1	2.07	4.67	7.39	13
outFreqSvm	75.15%	17.55%	0	0	0	0	1	1	1.46	4	4.67	5.44	13
tfidfAvg	73.56%	0.05%	1	1	1	1	1	1	1	1	3	6	13
heuristic3	**73.34%**	**2.02%**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**4**	**4**	**4**	**13**
heuristic1	**70.58%**	**8.09%**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**4**	**4.02**	**13**
outFreqRelCs	64.40%	0.19%	0	0	0	0	0	0	0	0	0	1.49	13
outFreqRelSum	62.52%	0.09%	0	0	0	0	0	0	0	1.64	2	2	13
outFreqRelRf	61.30%	0.12%	0	0	0	0	0.01	0.58	1	1	1	2	13
outFreqRelSvm	58.39%	0.17%	0	0	0	0	0	0	0	0	1.25	2	13
heuristic2	**56.85%**	**2.26%**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**13**
appearInAllDomains	50.00%	50.00%	0	0	0	0.01	0.02	0.03	0.08	0.17	0.33	0.83	13
freqDomnRatioMin	24.93%	1.12%	0	0	0	0	0	0	0	0	0	0	13

5.5 CrossBee Methodology Empowerment with Controlled Vocabularies

This section describes an extension of the CrossBee methodology, described in Sections 5.2 and 5.3, with a new ingredient: the use of a controlled vocabulary for improving B-term detection and ranking. The motivation for using predefined controlled vocabularies is to reduce the heuristic search space which, consequently, reduces the running times of B-term discovery algorithms. Furthermore, controlled vocabularies ensure consistency and resolve ambiguity inherent in normal human languages where the same concept can be given different names. In this way, they improve the quality and organization of retrieved knowledge, given that they consist of predefined, authorized terms that have been pre-selected by the designers of the vocabulary that are experts in the subject area. Controlled vocabularies solve problems of homographs and synonyms by a bijection between concepts and authorized terms.

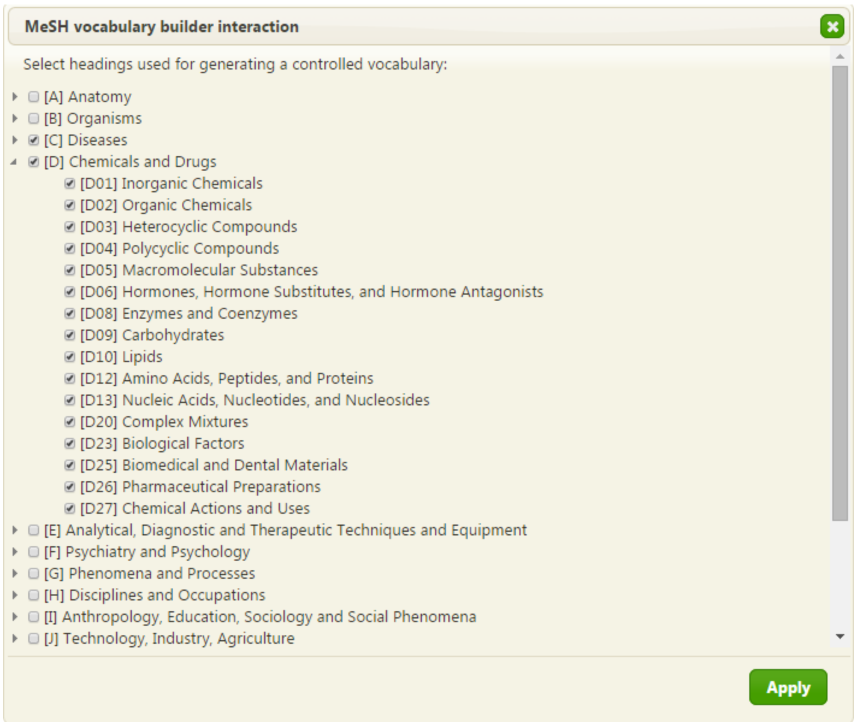
5.5.1 The MeSH Vocabulary

MeSH (Medical Subject Headings) is a controlled vocabulary used for indexing articles for PubMed, designed by The National Library of Medicine (NLM). Figure 5.11 shows a top-level example of the MeSH structure and hierarchy. The 2015 version of MeSH contains a total of 27,455 subject headings, also known as descriptors. Each descriptor is assigned a unique tree number (shown in square brackets in Figure 5.11) that facilitates search and filtering. Most of the descriptors are accompanied by a short description or definition, links to related descriptors, and a list of synonyms or very similar terms (known as entry terms). Because of these synonym lists MeSH can also be viewed as a thesaurus.

We have implemented a vocabulary construction tool called *MeSH filter* as an interac-

```
Nervous System Diseases [C10]
  Central Nervous System Diseases [C10.228]
    Brain Diseases [C10.228.140]
      Headache Disorders [C10.228.140.546]
        Headache Disorders, Primary [C10.228.140.546.399]
          Migraine Disorders [C10.228.140.546.399.750]
            Alice in Wonderland Syndrome [C10.228.140.546.399.750.124]
            Migraine with Aura [C10.228.140.546.399.750.250]
            Migraine without Aura [C10.228.140.546.399.750.450]
            Ophthalmoplegic Migraine [C10.228.140.546.399.750.725]
          Tension-Type Headache [C10.228.140.546.399.875]
          Trigeminal Autonomic Cephalalgias [C10.228.140.546.399.937]
```

Figure 5.11: Example of MeSH structure and hierarchy.



MeSH vocabulary builder interaction

Select headings used for generating a controlled vocabulary:

- [ ] [A] Anatomy
- [ ] [B] Organisms
- [✓] [C] Diseases
- [✓] [D] Chemicals and Drugs
  - [✓] [D01] Inorganic Chemicals
  - [✓] [D02] Organic Chemicals
  - [✓] [D03] Heterocyclic Compounds
  - [✓] [D04] Polycyclic Compounds
  - [✓] [D05] Macromolecular Substances
  - [✓] [D06] Hormones, Hormone Substitutes, and Hormone Antagonists
  - [✓] [D08] Enzymes and Coenzymes
  - [✓] [D09] Carbohydrates
  - [✓] [D10] Lipids
  - [✓] [D12] Amino Acids, Peptides, and Proteins
  - [✓] [D13] Nucleic Acids, Nucleotides, and Nucleosides
  - [✓] [D20] Complex Mixtures
  - [✓] [D23] Biological Factors
  - [✓] [D25] Biomedical and Dental Materials
  - [✓] [D26] Pharmaceutical Preparations
  - [✓] [D27] Chemical Actions and Uses
- [ ] [E] Analytical, Diagnostic and Therapeutic Techniques and Equipment
- [ ] [F] Psychiatry and Psychology
- [ ] [G] Phenomena and Processes
- [ ] [H] Disciplines and Occupations
- [ ] [I] Anthropology, Education, Sociology and Social Phenomena
- [ ] [J] Technology, Industry, Agriculture

Figure 5.12: *MeSH filter* is a widget in the TextFlows platform designed to help the user to quickly select the descriptors of interest.

tive widget in the TextFlows platform. This implementation uses synonym lists from the MeSH 2015 database, available online[7]. Figure 5.12 shows the interface to the developed interaction widget. The interface is designed to enable the selection of descriptors of interest from the hierarchy of descriptors. Its final output is a text file containing all the terms that belong to the user selected descriptors from the MeSH hierarchy.

5.5.2 Extended Methodology Workflow

This section describes how we have upgraded the proposed methodology with the ability to use a predefined controlled vocabulary for reduction of the B-term search space. This

[7]http://www.nlm.nih.gov/mesh/filelist.html

not only increases the efficiency of the heuristic calculation algorithms, but also tends to improve the relevance of top ranked B-terms due to reduced ambiguities in human languages. The upgraded methodology is shown in Figure 5.13. Compared to the initial methodology shown in Figure 5.6, the new workflow[8] includes two new steps: vocabulary acquisition and vocabulary preprocessing.

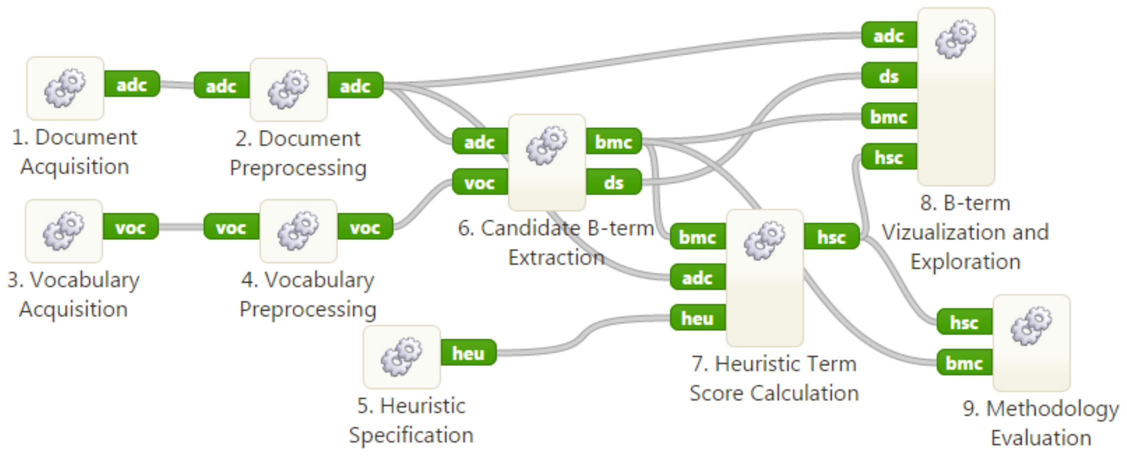

1. Document Acquisition
adc
2. Document Preprocessing
adc
adc
3. Vocabulary Acquisition
voc
4. Vocabulary Preprocessing
voc
voc
5. Heuristic Specification
heu
6. Candidate B-term Extraction
adc
voc
bmc
ds
7. Heuristic Term Score Calculation
bmc
adc
heu
hsc
8. B-term Vizualization and Exploration
adc
ds
bmc
hsc
9. Methodology Evaluation
hsc
bmc

Figure 5.13: Methodological steps of the extended cross-domain literature mining process. This workflow is publicly available at `http://textflows.perovsek.com/workflow/55/`.

A procedural explanation of the new steps of the upgraded workflow of Figure 5.13 is presented below.

Vocabulary Acquisition

Figure 5.14 shows a structurally simple methodological step of vocabulary acquisition that contains only two components. Its goal is to acquire the vocabulary and transform it into the *AnnotatedDocumentCorpus* data object, so it can be preprocessed afterwards. The components are responsible for the following tasks:

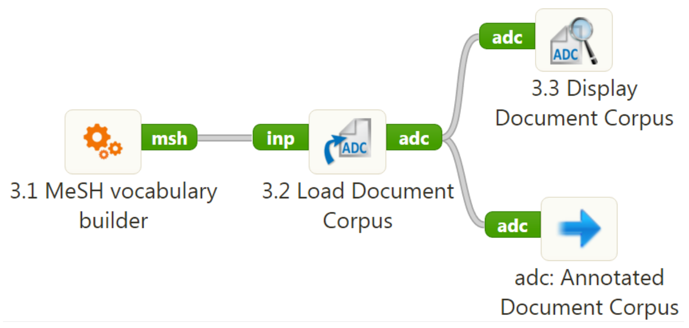

3.1 MeSH vocabulary builder
msh
inp
3.2 Load Document Corpus
adc
adc
3.3 Display Document Corpus
adc
adc: Annotated Document Corpus

Figure 5.14: Vocabulary acquisition workflow.

3.1. construct vocabulary using synonym lists from the MeSH 2015 database. The user is shown an interactive interface designed to enable the selection of descriptors of interest from the hierarchy of descriptors. The output is a text file containing all the terms that belong to the selected descriptors from the MeSH hierarchy.

[8]Publicly available at `http://textflows.perovsek.com/workflow/55/`

3.2. load vocabulary into *AnnotatedDocumentCorpus* data structure
3.3. optional check of vocabulary acquisition by visual inspection of the created corpus.

Vocabulary Preprocessing

The vocabulary acquisition step is followed by the vocabulary preprocessing step, which is itself a workflow implemented as shown in Figure 5.15.

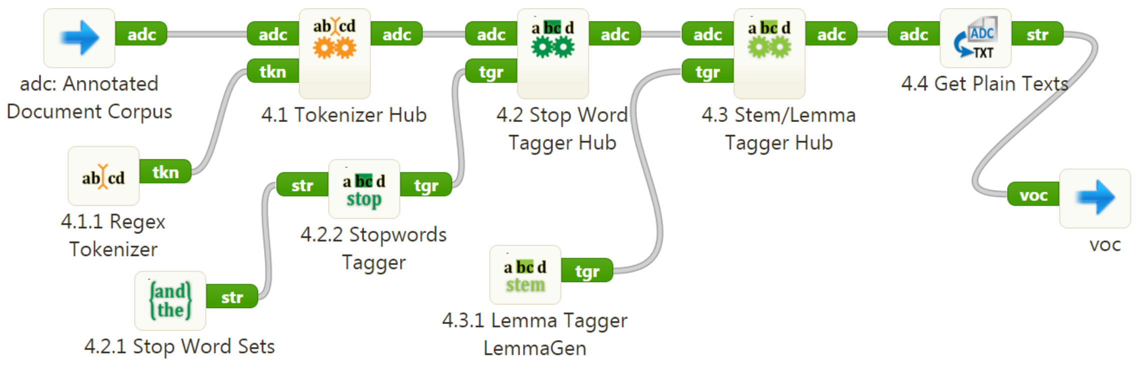


Figure 5.15: Vocabulary preprocessing workflow.

In order to ensure the proper matching between terms from the vocabulary and document corpus, we preprocessed the vocabulary using the preprocessing techniques, which were also used for preprocessing the document corpus in Step 2 and are described in more detail in Section A.1.2.

The main components here are tokenization, stopwords labeling and token stemming or lemmatization. The individual components perform the following tasks:

4.1. split vocabulary terms (represented as separate documents) to tokens (the basic units for further text processing),
   4.1.1. create a simple tokenizer object based on regular expressions,
4.2. tag stopword tokens by using a stopword tagger (component 4.2.2),
   4.2.1. load standard English stopwords,
   4.2.2. define the stopword tagger using the standard English stopwords only (the detected stopwords are discarded),
4.3. lemmatize tokens by applying the LemmaGen lemmatizer[9] [91],
   4.3.1. create an instance of LemmaGen lemmatizer,
4.4. construct the output of this vocabulary preprocessing step. The result is a Python list of strings, where every item represents a vocabulary term, constructed from lemmas and without stopwords.

Candidate B-term Extraction

After completing the preprocessing steps, the resulting whitelist output is used in Candidate B-term Extraction step for filtering out terms that are not part of the controlled vocabulary.

[9]LemmaGen is an open source lemmatizer with 15 prebuilt European lexicons. Its source code and documentation is publicly available at `http://lemmatise.ijs.si/`

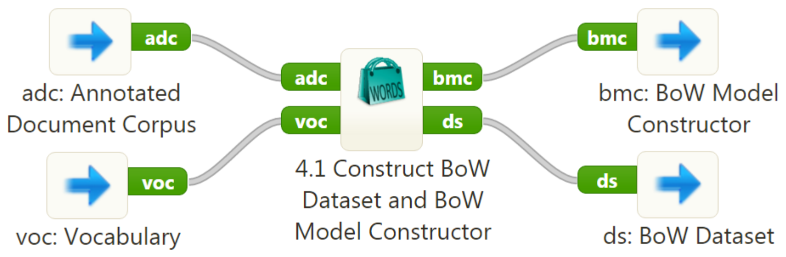

adc
adc: Annotated Document Corpus
voc
voc: Vocabulary
adc
voc
bmc
ds
WORDS
4.1 Construct BoW Dataset and BoW Model Constructor
bmc
bmc: BoW Model Constructor
ds
ds: BoW Dataset

Figure 5.16: Candidate B-term extraction.

The candidate B-term extraction step, shown as a workflow in Figure 5.16, is one of the core steps of the methodology. Even though it contains only one component, it has a very important and complex goal of transforming the inputted annotated document corpus into the BoW model. Consequently, the documents are represented in the form of feature vectors of term occurrences in the documents (for the purpose of visualization of documents and the need of highlighting and emphasizing of specific terms).

Moreover, the *Construct BoW Dataset and BoW Model Contructor* widget optionally accepts either a Python list or a file representing the controlled vocabulary with the objective of filtering out terms that are not part of the controlled vocabulary. Each line in the file (also possible: each string from a Python list of strings) can be in one of the two following formats:

- *One term per line*: every single line in the text file represents one separate term. Only terms which appear in this file are later used in construction of the BoW space, and thus in the next steps of the methodology.

- *Synonym format*: additionally, term synonyms are listed after the term, separated by commas.

  $$\text{term}_1 \rightarrow \text{synonym}_{1a}, \text{synonym}_{1b}...$$

  Every synonym in the document corpus is then substituted with the term, which appears at the first position in the corresponding line.

Another task of this step is to capture the exact parsing procedure, which is needed in order to perform various computations which are performed in the advanced heuristic term scores calculation step. The outputted *BowModelContructor* object also contains the final vocabulary of all terms.

Having described the developed methodology, the following section presents the results of methodology evaluation in more detail.

5.5.3 Experiments and Results

This section presents the evaluation of the extended methodology for literature-based discovery. Similar as in Section 5.4.6, we applied different base and ensemble heuristics on two problems: the standard migraine-magnesium literature mining benchmark problem [63], and the autism-calcineurin dataset, originated by [68] and [69]. In both cases, we compared our extended methodology to the methodology proposed by Juršič et al., which is described in Section 5.2.1.

Results on the Migraine-Magnesium Dataset

In this section we demonstrate that by using a predefined controlled vocabulary we can increase the heuristics' capabilities to rank the B-terms at the beginning of the term list. We have repeated the experiments on the migraine-magnesium domain, described in Section 5.4.6, except that we now used a predefined vocabulary constructed from MeSH using the "MeSH filter" widget. As we were particularly interested in the bridging terms between migraine (a disease) and magnesium (a chemical element) as well as the circumstances and processes observed between them, we only selected categories [C] *Diseases*, [D] *Chemicals and drugs* and [G] *Phenomena and Processes* from the MeSH hierarchy. In the experiment we used the workflow shown in Figure 5.13. The generated vocabulary was used in the candidate B-term extraction step as a whitelist filter. As a result, not all B-terms identified by Swanson [63] were part of the used controlled vocabulary (shown in Table 5.6).

Table 5.6: B-terms for the migraine-magnesium dataset identified by Swanson [63]. The 17 terms which are crossed out were not part of the used controlled vocabulary, therefore heuristics were unable to identify them as B-term candidates.

1 5 ht	~~16 convulsive~~	31 prostaglandin
2 5 hydroxytryptamine	~~17 coronary spasm~~	~~32 prostaglandin e1~~
3 5 hydroxytryptamine receptor	18 cortical spread depression	33 prostaglandin synthesis
~~4 anti aggregation~~	19 diltiazem	~~34 reactivity~~
5 anti inflammatory	20 epilepsy	35 seizure
6 anticonvulsant	~~21 epileptic~~	36 serotonin
~~7 antimigraine~~	~~22 epileptiform~~	37 spasm
~~8 arterial spasm~~	23 hypoxia	~~38 spread~~
~~9 brain serotonin~~	24 indomethacin	39 spread depression
10 calcium antagonist	~~25 inflammatory~~	~~40 stress~~
~~11 calcium blocker~~	26 nifedipine	41 substance p
12 calcium channel	~~27 paroxysmal~~	~~42 vasospasm~~
13 calcium channel blocker	28 platelet aggregation	43 verapamil
14 cerebral vasospasm	~~29 platelet function~~	
15 convulsion	30 prostacyclin	

The results of the methodology using a controlled vocabulary on the migraine-magnesium domain are presented in Table 5.8. The comparison of the heuristics' capabilities to rank the B-terms at the beginning of the term list in the migraine-magnesium domain from Table 5.7 and Table 5.8 shows an advantage of using the controlled vocabulary. By inspecting the number of B-terms found by the base heuristics in the ranked first n terms, we notice that using the controlled vocabulary in the migraine-magnesium domain resulted in a much higher concentration of Swanson's B-terms among the best ranked terms. Even though the heuristics based on banded matrices were already among the best performing heuristics, the reduced search space affected them the most. Not only the running times of the bidirectional MBA algorithm decreased when using the controlled vocabulary, but also the relevance of top ranked B-terms increased, making the banded matrix-based heuristics to even outperform the ensemble heuristic without the vocabulary. The full list of base heuristics and their scores is available in Appendix A.2.

A consequence of the improved ranking capabilities of the base heuristics, is the increased quality of the best ranked terms of the ensemble as well. The ensemble heuristic, described in Section 5.2.3, performing ensemble voting of six elementary heuristics, also resulted in very favorable results on the migraine-magnesium domain, where two additional B-terms were found among the first 10 terms and 7 more among the first 100 terms.

As explained in Section 5.5 a predefined controlled vocabulary can greatly reduce the

Table 5.7: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the migraine-magnesium dataset.

Heuristic Name	Number of B-terms among the top n ranked terms								
	5	10	20	50	100	200	500	1,000	2,000
heuristic3	0	0	0	1	3	9	14.92	28.48	43
heuristic2	0	0	0	1	6	7.91	12.65	29.62	43
heuristic1	0	0	0	1	2	4.48	16.5	26.58	43
outFreqSum	0	0	0	0	0	2	14.28	27.11	43
outFreqRelSvm	0.12	0.24	0.48	1	1.63	5.92	14.69	29	43
outFreqRelRf	0.24	0.48	0.95	1	3.72	7.92	15.8	30.33	43
outFreqRelCs	0	0	1	1	2.69	5.07	11	27	43
tfidfDomnSum	0	0	0	0	1	2	11	26.14	43
freqRatio	0	0	1	1	4	5	11.65	23.09	43
ensemble	1	1	1	5	6	9	18.57	28	43

Table 5.8: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the migraine-magnesium dataset using a controlled vocabulary.

Heuristic Name	Number of B-terms among the top n ranked terms								
	5	10	20	50	100	200	500	1,000	2,000
heuristic3	0	2	6	7	14	22.68	26	26	26
heuristic2	2	5	5.43	12	15.5	25	26	26	26
heuristic1	1	2	2.25	9	13.63	20.15	26	26	26
outFreqSum	0	2	3	5.45	15.05	16.96	26	26	26
outFreqRelSvm	0.45	0.91	1.82	3.25	9	15	26	26	26
outFreqRelRf	0.56	1.11	2	5	8	15	26	26	26
outFreqRelCs	0.31	0.63	1	5	7.06	14	26	26	26
tfidfDomnSum	0	1	1	4	10	19	26	26	26
freqRatio	0	1	1	2	5.96	14.56	26	26	26
ensemble	1	3	4	9	13	19	26	26	26

B-term search space. As a side effect, we were unable to: (a) perform AUROC evaluation comparison due to different number of terms in the vocabulary; as a result, Table 5.8 provides only evaluation which lists the number of B-terms found in the ranked first n terms, (b) detect all B-terms, identified by Swanson (the crossed out B-terms in Table 5.6 were not part of the used controlled vocabulary); this could be solved using larger controlled vocabularies, though we must be careful not to overfit the vocabulary to the expected results.

Results on the Autism-Calcineurin Dataset

In this section we replicated the experiments, described in the previous section, using a predefined controlled vocabulary on the autism-calcineurin dataset. Similarly, we wanted to increase the heuristics' capabilities (from the workflow, illustrated in Figure 5.13) to rank the B-terms at the beginning of the term list. We used the same predefined vocabulary as with the migraine-magnesium domain, which was constructed from MeSH using the following categories: [C] *Diseases*, [D] *Chemicals and drugs* and [G] *Phenomena and Processes* were used for building the controlled vocabulary. Table 5.9 shows the list of B-terms identified by Petrič et al. [69] that were also part of the used controlled vocabulary.

Inspecting the heuristics' capabilities to rank the B-terms at the beginning of the term list in the autism-calcineurin domain (Table 5.10 and Table 5.11) shows again the advantage of using a controlled vocabulary. The increase in the number of B-terms found

Table 5.9: B-terms for the autism-calcineurin dataset identified by Petrič et al. [69]. The four terms which are crossed out were not part of the used controlled vocabulary, therefore heuristics were unable to identify them as B-term candidates.

~~1 synaptic~~	6 bcl 2	~~11 22q11 2~~
2 synaptic plasticity	7 type 1 diabetes	~~12 maternal hypothyroxinemia~~
3 calmodulin	8 ulcerative colitis	13 bombesin
4 radiation	9 asbestos	
~~5 working memory~~	10 deletion syndrome	

in the ranked first n terms when using the controlled vocabulary is even more significant than in the migraine-magnesium domain. Similarly to the migraine-magnesium dataset the most improved heuristics were banded matrix-based heuristics. Even though in this domain they do not outperform the ensemble, heuristic 3 managed to rank one B-term among the top 5 ranked terms (before only among the top 500).

The ensemble heuristic extended with the controlled vocabulary also finds the first B-term among the top 5 ranked terms (before only among the top 10) and the second B-term among the top 50 ranked terms (before only among 200). These results confirm the findings that controlled vocabularies can increase the heuristics' capacities to rank the B-terms at the beginning of the term list and, thus, provide a more efficient exploration task to the end-user of the platform.

Table 5.10: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the autism-calcineurin dataset.

Heuristic Name	Number of B-terms among the top n ranked terms										
	5	10	20	50	100	200	500	1,000	2,000	5,000	all
heuristic3	0	0	0	0	0	0	1	4	4	4	13
heuristic2	0	0	0	0	0	0	0	0	0	1	13
heuristic1	0	0	0	0	0	0	0	1	4	4.02	13
outFreqSum	0	0	0	0	0	1	1	3	5	7	13
outFreqRelSvm	0	0	0	0	0	0	0	0	1.25	2	13
outFreqRelRf	0	0	0	0	0.01	0.58	1	1	1	2	13
outFreqRelCs	0	0	0	0	0	0	0	0	0	1.49	13
tfidfDomnSum	0	0	0	0	1	1	2	2	4	5	13
freqRatio	1	1	1	1	1	1	1	3	5	8.99	13
ensemble	0	1	1	1	1	2	3	4	6	8	13

Table 5.11: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the autism-calcineurin dataset using a controlled vocabulary.

Heuristic Name	Number of B-terms among the top n ranked terms										
	5	10	20	50	100	200	500	1000	2,000	5,000	all
heuristic3	1	1	1	1	1	1	1	4	6	9	9
heuristic2	0	0	0	1	1	1	2	2	5.76	9	9
heuristic1	0	1	1	1	1	2	2	5	7	9	9
outFreqSum	0	0	0	0	0	4	5.85	7	8	9	9
outFreqRelSvm	0	0	1	1	1	1	2	4	9	9	9
outFreqRelRf	0	0	0	0	0	1	1	3	8	9	9
outFreqRelCs	0	0	0	0	0	0	2	3	7	9	9
tfidfDomnSum	0	1	1	1	1	2	3	7	9	9	9
freqRatio	1	1	1	1	2	3	3.6	6.01	9	9	9
ensemble	1	1	1	2	2	2	4	6	8	9	9

Tables 5.10 and 5.11 show only scores for the ensemble heuristic and base heuristics included in the ensemble (as identified by Juršič et al. [19], [70]). The full list of base heuristics and their scores is available in Appendix A.3.

To support bisociative cross-domain literature mining, the TextFlows platform includes implementations of several elementary and ensemble heuristics that guide the expert in the process of exploring new cross-context bridging terms. We have extended the TextFlows platform with several components, which—together with document exploration and visualization features of the CrossBee human-computer interface—make it a powerful, user-friendly text analysis tool for exploratory cross-domain knowledge discovery. We also propose a further extension of the methodology by facilitating the use of controlled vocabularies, enhancing the heuristics capability to rank the actual B-terms at the top of the ranked term list. Results show that using a predefined controlled vocabulary not only increases the efficiency of the heuristic calculation algorithms, but also tends to improve the relevance of top ranked B-terms. Consequently, the described approach enables the user to perform the exploration task more effectively, potentially leading to new discoveries.

Although not primarily designed for this task, we show in Appendix B that our methodology can also be used for creating pairs of sentences from different domains, which combine into surprising, funny or even insightful pieces of text when put together and considered as a whole. Bridging terms, appearing in both sentences, detected by our methodology function as a kind of glue, contributing to the coherency and increasing the potential for combinations to be meaningful.

With all these features, the TextFlows platform (with its reusable text analytics workflows) combined with the CrossBee document exploration interface, has become a publicly available creativity support tool (CST), supporting creative discovery of new cross-domain hypotheses.

Chapter 6

Relational Data Mining Scenario

This chapter presents a propositionalization technique called *wordification* which can be seen as a transformation of a relational database into a corpus of text documents. Wordification constructs simple, easy to understand features, acting as words in the transformed Bag-of-Words representation. The main advantages of the approach are: simple implementation, accuracy comparable to competitive methods, and greater scalability, as it performs several times faster on all experimental databases. Furthermore, the wordification methodology and the evaluation procedure are implemented as executable workflows in the web-based text mining platform TextFlows. The implemented workflows include also several other ILP and RDM algorithms as well as the utility components that were added to the platform to enable access to these techniques to a wider research audience.

The chapter is organized as follows. Section 6.1 defines the problem and gives motivation for developing new propositionalization technique inspired by text mining. Section 6.2 gives an informal overview of the wordification methodology, while Section 6.3 presents the formalism and the details of the developed wordification algorithm. The implementation of the methodology as a workflow in the TextFlows platform is described in Section 6.4. Section 6.5 presents the evaluation methodology implementation and the experimental results.

6.1 Problem Definition and Motivation

Standard propositional data mining algorithms, included in established data mining tools like Weka [29], induce models or patterns learned from a single data table. On the other hand, the aim of Inductive Logic Programming (ILP) and Relational Data Mining (RDM) is to induce models or patterns from multi-relational data [20]–[23]. Most types of propositional models and patterns have corresponding relational counterparts, such as relational classification rules, relational regression trees or relational association rules.

For multi-relational databases in which data instances are clearly identifiable (the so-called individual-centered representation [73], characterized by one-to-many relationships among the target table—which contains the target variable—and other data tables), various techniques can be used for transforming a multi-relational database into a propositional single-table format [79]. After performing such a transformation [24], named *propositionalization* [25], standard propositional learners can be used, including decision tree and classification rule learners.

Inspired by text mining, we propose a new propositionalization approach to relational data mining, called *wordification*. Unlike other propositionalization techniques [24], [25], [75], [76], which first construct complex relational features (constructed as a chain of joins of one or more tables related to the target table), used as attributes in the resulting tabular data representation, wordification generates much simpler features with the aim of achieving greater scalability.

Wordification can be viewed as a transformation of a relational database into a set of feature vectors, where each original instance is transformed into a-kind-of 'document' represented as a Bag-of-Words (BoW) vector of weights of simple features, which can be interpreted as 'words' in the transformed BoW space. The 'words' constructed by wordification correspond to individual attribute-values of the target table and of the related tables, subsequently weighted by their Term Frequency-Inverse Document Frequency (TF-IDF) value [88], [111] (requiring real-valued attributes to be discretized first). Alternatively, instead of TF-IDF, simpler schemes can be used, such as term frequency (TF) 'word' count, or the binary scheme indicating just the presence/absence of a 'word' in the 'document'.

To intuitively phrase the main idea of wordification, take two simple examples illustrating the wordification data preprocessing step in class-labeled data, where each structured data instance is transformed into a tuple of simple features, which are counts/weights of individual attribute-value pairs. Take the well-known relational domain of East-West Trains [74] with cars containing different loads: one of the train's features in the BoW representation is the count/weight of rectangular loads it carries, no matter in which cars these loads are stored. Or in the standard Mutagenesis domain [112], a molecule may prove to be toxic if it contains a lot of atoms characterized by the property *atom_type=lead*, no matter how these atoms are bonded in the molecule. The main hypothesis of the wordification approach is that the use of this simple representation bias is suitable for achieving good results in classification tasks. Moreover, when using a binary scheme, this representation bias allows for simple and very intuitive interpretation in descriptive induction tasks, such as association rule learning from unlabeled multi-relational data. We describe one such task in Appendix D, where we illustrate the utility of wordification in a descriptive induction setting of learning association rules from two real-life domains, using data from a subset of the IMDB movies database and from a database of traffic accidents.

Wordification suffers from some loss of information, compared to propositionalization methods which construct complex first-order features (which get values *true* or *false* for a given individual) as a chain of joins of one or more tables related to the target table. Nevertheless, despite some information loss, wordification has numerous advantages. Due to the simplicity of features, the generated hypotheses are easily interpretable by domain experts. The feature construction step in wordification is very efficient, therefore it can scale well for large relational databases. As wordification constructs each 'document' independently from the other 'documents', a large main table can be divided into smaller batches of examples, which can be propositionalized in parallel. Next, wordification can use TF or TF-IDF word weighting to capture the importance of a given feature (attribute value) of a relation in an aggregate manner, while feature dependence is modelled by constructing a-kind-of word 'n-grams' as conjuncts of a predefined number of simple features. Finally, the wordification approach has the advantage of using techniques developed in the text mining community, such as efficient document clustering or word cloud visualization, which can now be effectively exploited in multi-relational data mining.

6.2 Informal Description of the Wordification Approach

This section provides an informal description of the proposed approach, where wordification is illustrated on a simplified variant of the well-known East-West Trains problem [74].

The transformation from a relational database representation into a Bag-of-Words feature vector representation is illustrated in Figure 6.1, where the input to wordification is a relational database, and the output is a set of feature vectors, which can be viewed as a

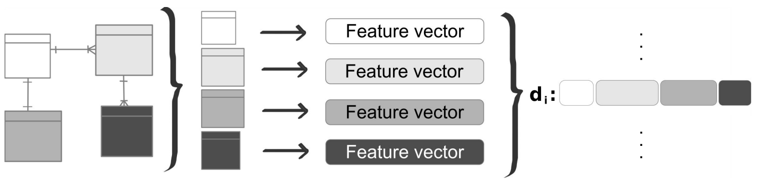


Figure 6.1: The transformation from a relational database representation into a Bag-of-Words feature vector representation. For each individual entry of the main table one Bag-of-Words (BoW) vector d_i of weights of 'words' is constructed, where 'words' correspond to the features (attribute values) of the main table and the related tables.

corpus of text documents represented in the Bag-of-Words (BoW) vector format. Each text document represents an individual entry of the main data table. A document is described by a set of words (or features), where a word is constructed as a combination of the table name, name of the attribute and its discrete (or discretized) value[1]:

$$[table\ name]\_[attribute\ name]\_[value]. \tag{6.1}$$

Such constructs are called *word-items* or *witems* or simply *words* in the rest of the thesis. Note that values of every non-discrete attribute need to be discretized beforehand in order to be able to represent them as word-items. For each individual, the word-items are first generated for the main table and then for each entry from the related tables, and finally joined together according to the relational schema of the database[2]. In the described transformation there is some loss of information as a consequence of building the document for each instance (each individual row in the main table) by concatenating all word-items from multiple instances (rows) of the connected tables into a single document. To overcome this loss, we extended the document construction step of the initial wordification methodology by concatenating to the document also n-grams of word-items, constructed as conjunctions of several word-items. These concatenations of elementary word-items represent conjunctions of features occurring together in individual instances (rows of joined tables). Technically, n-gram construction is performed by taking every combination of length k of word-items from the set of all word-items corresponding to the given individual, and concatenating them as follows:

$$[witem_1]\_\_[witem_2]\_\_\ \dots\ \_\_[witem_k], \tag{6.2}$$

where $1 \leq k \leq n$ and—as mentioned earlier—each word-item is a combination of the table name, name of the attribute and its discrete value. The witems are concatenated in a predetermined order, each using the "__" concatenation symbol.

In the rest of this chapter, for simplicity, we refer to individuals as documents, to features as words, and to the resulting representation as the Bag-of-Words (BoW) representation. For a given word w in document d from corpus D, the TF-IDF measure is defined as follows:

$$\text{tfidf}(w, d) = \text{tf}(w, d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{6.3}$$

where tf($\cdot$) represents the number of times word w appears in document d. In other words, a word with a high TF-IDF value will be considered important for the given individual provided that it is frequent within this document and not frequent in the entire corpus.

[1]See Line 4 in Algorithm 6.2 presented in Section 6.3.3.
[2]See Line 10 in Algorithm 6.2 presented in Section 6.3.3.

Consequently, the weight of a word provides a strong indication of how relevant is the feature for the given individual. The TF-IDF weights can then be used either for filtering out words with low importance or using them directly by a propositional learner.

In addition to the TF-IDF weighting scheme, the implementation of wordification (described in detail in Section 6.4) includes also the term frequency (TF) and the binary (0/1) weighting schemes. A comparison of the three schemes can be found in the Appendix C (see Table C.1). Given that different weighting schemes do not perform significantly differently on the classification tasks used in our experiments, in the rest of the paper we use the TF-IDF scheme since this form of weighting is prevalent in text mining applications.

TRAIN

trainID	eastbound
t1	east
...	...
t5	west
...	...

CAR

carID	shape	roof	wheels	train
c11	rectangle	none	2	t1
c12	rectangle	peaked	3	t1
...	...	...	...	...
c51	rectangle	none	2	t5
c52	hexagon	flat	2	t5
...	...	...	...	...

Figure 6.2: Example input for wordification in the East-West Trains domain.

```
t1: [car_roof_none, car_shape_rectangle, car_wheels_2,
car_roof_none__car_shape_rectangle, car_roof_none__car_wheels_2,
car_shape_rectangle__car_wheels_2, car_roof_peaked, car_shape_rectangle,
car_wheels_3, car_roof_peaked__car_shape_rectangle,
car_roof_peaked__car_wheels_3, car_shape_rectangle__car_wheels_3], east
...
t5: [car_roof_none, car_shape_rectangle, car_wheels_2,
car_roof_none__car_shape_rectangle, car_roof_none__car_wheels_2,
car_shape_rectangle__car_wheels_2, car_roof_flat, car_shape_hexagon,
car_wheels_2, car_roof_flat__car_shape_hexagon,
car_roof_flat__car_wheels_2, car_shape_hexagon__car_wheels_2], west
...
```

Figure 6.3: The database from Figure 6.2 in the Bag-of-Words document representation.

The wordification approach is illustrated on a modified and substantially simplified version of the well-known East-West Trains domain [74], where the input database consists

id	car_shape _rectangle	car_roof _peaked	car_wheels_3	car_roof_peaked__ car_shape_rectangle	car_shape_rectangle __car_wheels_3	...	class
t1	0.000	0.693	0.693	0.693	0.693	...	east
...	...	...	...	...	...	...	...
t5	0.000	0.000	0.000	0.000	0.000	...	west
...	...	...	...	...	...	...	...

Figure 6.4: The transformed database (consisting of TF-IDF values, which are zero if the term appears in all documents) from Figure 6.2 using the wordification approach. This final output can be given as an input to a propositional classifier.

of just two tables shown in Figure 6.2, where we have only one East-bound and one West-bound train, each with just two cars with certain properties. Note that in the experimental section we use the standard version of the East-West Trains domain.

The TRAIN table is the main table and the trains are the individuals. We want to learn a classifier to determine the direction of an unseen train. For this purpose the direction attribute is not preprocessed and is only appended to the resulting feature vector (list of words).

First, the corresponding two documents (one for each train t1 and t5) are generated, as shown in Figure 6.3. After this, the documents are transformed into the Bag-of-Words representation by calculating the TF-IDF values for each word of each document (using Equation 3.1) with the class attribute column appended to the transformed Bag-of-Words table, as shown in Figure 6.4. For simplicity, only unigrams and bigrams are shown in this example.

6.3 Wordification Methodology

This section formally describes the wordification methodology by presenting the input data model and input language bias, the relational database representation, followed by the presentation of the pseudo-code and the worst-case complexity analysis of the wordification algorithm.

6.3.1 Data Model

A data model describes the structure of the data. It can be expressed as an entity-relationship (ER) diagram. The ER diagram, illustrated in Figure 6.5, shows three relations appearing in the original East-West Trains problem (in addition to the TRAIN and CAR relationship, it includes also the LOAD relationship, which was skipped for simplicity in Figure 6.2). The boxes in the ER diagram indicate *entities*, which are individuals or parts of individuals. Here, the Train entity is the individual, each Car is part of a train, and each Load is part of a car. The ovals denote attributes of entities. The diamonds indicate *relationships* between entities. There is a *one-to-many relationship* from Train to Car, indicating that each train can have an arbitrary number of cars but each car is contained in exactly one train; and a *one-to-one relationship* between Car and Load, indicating that each car has exactly one load and each load is part of exactly one car.

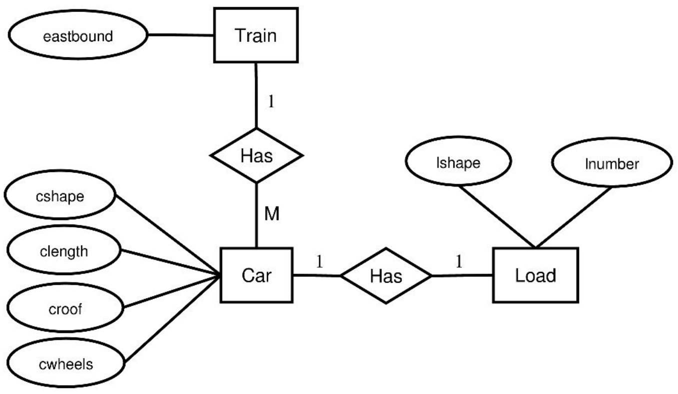


Figure 6.5: Entity-relationship diagram for the East–West challenge.

Entity-relationship diagrams can be used to choose a proper logical representation for the data. If we store the data in a relational database the most obvious representation is to have a separate table for each entity in the domain, with relationships being expressed by *foreign keys*.[3] This is not the only possibility: for instance, since the relationship between Car and Load is one-to-one, both entities could be combined in a single table, while entities linked by a one-to-many relationship cannot be combined without either introducing significant redundancy or significant loss of information, e.g., introduced through aggregate attributes. Note that fixed number of arguments (and consequently one-to-many relationships between multiple tables) distinguish relational learning and inductive logic programming from propositional learning.

In wordification, we use the entity-relationship diagram to define types of objects in the domain, where each entity will correspond to a distinct type. The data model constitutes a *language bias* that can be used to restrict the hypothesis space and guide the search. In most problems, only individuals and their parts exist as entities, which means that the entity-relationship model has a tree-structure with the individual entity at the root and only *one-to-one* or *one-to-many* relations in the downward direction. Representations with this restriction are called *individual-centered representations.* This restriction determines the language bias, constraining the relational database input to wordification.

6.3.2 Formal Setting

The framework, established in this section, defines a learning setting which is very similar to the standard propositionalization problem setting. As in every propositionalization approach to relational data mining, a two-step approach is implemented: (1) in the first propositionalization step the data is transformed from a relational database format to a tabular format, and (2) the tabular data is used as input for learning models or patterns by a selected propositional learner, having its own hypothesis language bias (e.g., decision trees or propositional classification rules). The formal framework described below focuses only on step (1) of the two-step wordification methodology. For simplicity, the formalization describes the setting using only unigram features.

Input

The input to wordification is a *relational database* (RDB), given as a set of relations $\{R_1, ..., R_n\}$ and a set of foreign-key connections between the relations denoted by $R_i \to R_j$, where R_i has a foreign-key pointing to relation R_j. The foreign-key connections correspond to the relationships in the entity-relationship diagram. For example, the `train` attribute in the CAR relation is a foreign-key referring to `trainID` in TRAIN. It defines the `CAR` $\to$ `TRAIN` connection; as expected, it is a many-to-one connection from CAR to TRAIN.

A n-ary relation R_i is formally defined as a set of *tuples*: a subset of the Cartesian product of m_i domains: $R_i \subset \prod_{j=1}^{m_i} D_{i_j} = D_{i_1} \times D_{i_2} \times \ldots \times D_{i_{m_i}}$, where a *domain* (or a *type*) is a specification of the valid set of values for the corresponding argument.

$$D_{i_j} = \{v_{i_{j_1}}, v_{i_{j_2}}, \ldots, v_{i_{j_{k_{ij}}}}\}$$

Note that for wordification we require that each domain D_{i_j} must have a finite number of unique values k_{ij}, thus discretization of continuous domains is needed.

A further requirement is that the RDB must be individual-centered. This means that a target relation $R_T \in$ RDB must exist, such that it does not have any foreign keys:

$$\nexists i : R_T \to R_i; R_i \in \text{RDB}$$

[3]In the context of relational databases, a foreign key is a field in a relational table that matches a candidate key of another table. The foreign key can be used to cross-reference tables.

Output

Having established the data model, the individual-centered data representation language bias and the relational database representation of input data, the formal output (a transformed single-relation representation $R_{T'}$) of the wordification methodology can be defined as follows:

$$R_{T'} \subset \prod_{i,j,k} D_{T'_{ijk}} = \prod_{i,j,k} \mathrm{domain}(R_i, D_{i_j}, v_{i_{j_k}}); R_i \xrightarrow{*} R_T$$

or in other words, one domain in the resulting relation $R_{T'}$ is defined for each relation R_i (that is connected by following the foreign-key path, denoted by $\xrightarrow{*}$ to R_T), and each of its domains D_{i_j} as well as domain values $v_{i_{j_k}}$. These domains have the property

$$D_{T'_{ijk}} = \mathbb{R}_0^+$$

since they are determined by the TF-IDF formula. This final output relation (table) can be given as an input to any propositional learner.

6.3.3 Wordification Algorithm

This section presents the wordification methodology by describing in detail the individual transformation steps in Algorithm 6.1 and Algorithm 6.2.

The algorithm starts recursive document construction on the instances of the main table (Lines 3-7 in Algorithm 6.1). First it creates word-items for the attributes of the target table (Lines 2-6 in Algorithm 6.2), which is followed by concatenations of the word-items and results of the recursive search through examples of the connecting tables (Lines 8-16 in Algorithm 6.2). As this document construction step is done independently for each example of the main table, this allows simultaneous search along the tree of connected tables. In order to perform concurrent propositionalization, Lines 3-7 in Algorithm 6.1 need to be run in parallel. A common obstacle in parallel computing is memory synchronization between the different subtasks, which is not the case here as concurrent processes in our implementation of wordification only need to share a cached list of subtrees. This list stores the results of subtree word concatenations in order to visit every subtree only once.

As wordification can produce a large number of features (words), especially when the maximal number of n-grams per word-items is large, we perform pruning of words that occur in less than a predefined percentage (5% on default) of documents. This reduces the

Algorithm 6.1: Wordification's *main(T,p,k)* procedure.

```
Input  : target table $T$, pruning percentage $p$, max number of witems per word $k$
Output: Propositionalized table $R$ with TF-IDF values, corpus of documents $D$
$D \leftarrow []$;
$W \leftarrow \emptyset$ ;                                   // vocabulary set
for $ex \in T$ do
    $d \leftarrow$wordify($T, ex, k$) ;                      // construct the document
    $D \leftarrow D + [d]$ ;                            // append document to the corpus
    $W \leftarrow W \cup keys(d)$ ;
end
$W \leftarrow$prune($W, p$) ;                              // optional step
return [ calculateTFIDFs($D, W$), $D$];
```

Algorithm 6.2: Wordification's *wordify(T,ex,k)* procedure.

```
Input  : table $T$, example $ex$ from table $T$, maximal number of witems per word $k$
Output: document word count $d$

$d \leftarrow \{\}$;                              // hash with a default value 0
for $i \leftarrow 1$ to $k$ do                    // for every word witem length
    for $comb \in$ attrCombs($ex, k$) do      // attr.  combinations of length k
        $d$[word($comb$)] $\leftarrow d$[word($comb$)] $+ 1$;
    end
end

// for every connected table through an example
for $secTable \in$ connectedTables($T$) do
    for $secEx \in secTable$ do
        if primaryKeyValue($ex$)=foreignKeyValue($secEx$) then
            for $(word, count) \in$ wordify($secTable, secEx, k$) do
                $d[word] \leftarrow d[word] + count$;
            end
        end
    end
end
return $d$;
```

size of trees by removing sections of the tree that is expected to provide little power for instance classification.

The constructed features are simple, and as we do not explicitly use existential variables in the new features (words), we instead rely on the Term Frequency-Inverse Document Frequency (TF-IDF) measure to implicitly capture the importance of a word for a given individual. In the context of text mining, TF-IDF value reflects how representative is a certain feature (word) for a given individual (document).

6.3.4 Time and Space Complexity

This section covers the worst-case complexity analysis of the Wordification algorithm. Let t be the number of tables in a database. To simplify the analysis, we assume that each table is connected with exactly one other table in a one-to-many relation. Let m_i and n_i be the number of rows and the number of attributes in table i, respectively. Further, let $m = \max(m_1, m_2, \ldots, m_t)$ and $n = \max(n_1, n_2, \ldots, n_t)$. The maximal number of rows is generally much higher than the number of attributes and the number of tables in a relational database: $t \ll n \ll m$. Let k be the maximal number of n-grams per word.

Time Complexity

The upper-bound time complexity for the propositionalization step of the wordification methodology when each word is constructed from one witem ($k = 1$) is $\mathcal{O}(m \cdot n \cdot m^{t-1}) = \mathcal{O}(n \cdot m^t)$. When we use words that are combinations of up to $1 \leq k \leq n$ witems (Lines 3-5 in Algorithm 6.2) the complexity of the algorithm is $\mathcal{O}(m^t \cdot \sum_{i=1}^{k} \binom{n}{i})$. As $\lim_{k \to n} \sum_{i=1}^{k} \binom{n}{i} = 2^n - 1$, the worst-case time complexity is therefore $\mathcal{O}(2^n \cdot m^t)$.

Space Complexity

The space complexity for the wordification algorithm using unigram features ($k = 1$) is $\mathcal{O}(m \cdot t \cdot n)$. When we increase the maximal word length (number of witems per word) the feature space of the algorithm also increases exponentially. When the maximal word length is equal to the maximal number of attributes, the space complexity is $\mathcal{O}(m \cdot t \cdot \sum_{i=1}^{k} \binom{n}{i})$. Following a similar reasoning as in the time complexity analysis, the worst case space complexity of the algorithm is therefore $\mathcal{O}(m \cdot t \cdot 2^n)$.

When $k \to n$ both time and space complexity are in its worst cases exponential in the number of attributes, but as evidenced from the experiments in Section 6.5, good performance can be achieved with $k = 1$ or $k = 2$, in which case the space complexity is linear $\mathcal{O}(m \cdot t \cdot n)$ and the time complexity is polynomial $\mathcal{O}(n \cdot m^t)$. Since t is usually small, the approach can perform orders of magnitude faster than its competitors, as demonstrated in Section 6.5.

6.4 Implementation

This section describes the implementation of the wordification methodology in the TextFlows platform. Figure 6.6 shows the entire workflow of the wordification methodology, which is composed of the *wordification* component from the ILP module as well as components from other modules of the TextFlows platform. The implementation allows the user to provide as input a relational database by connecting to a MySQL database server. First, the user is required to select the target table from the initial relational database, which will later represent the main table in the Wordification component of the workflow. Second, the user is able to discretize each table using one of the various discretization techniques provided. These discretized tables are used by the Wordification widget, where the transformation from the relational tables to a corpus of documents is performed. The workflow components are described in more detail below.

1. *MySQL Connect:* Since relational data is often stored in SQL databases, we use the MySQL package to access the training data by connecting to a MySQL database server. The MySQL Connect widget is used for entering information required to connect to a database (e.g., user credentials, database address, database name, etc.) in order to retrieve the training data from a MySQL database server and automatically construct the background knowledge and the training examples.

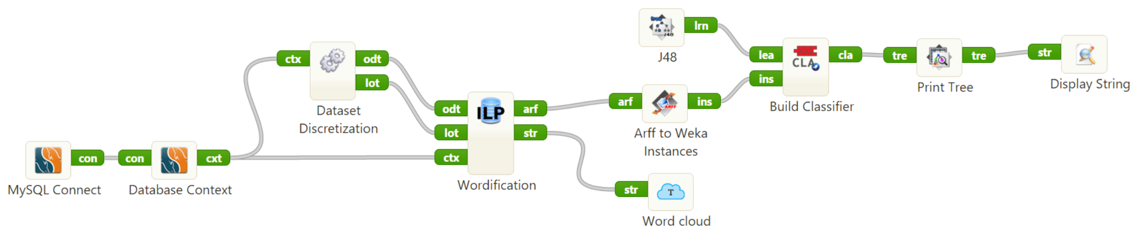


Figure 6.6: TextFlows wordification workflow with additional analyses after the wordification process. This workflow is publicly available at `http://textflows.perovsek.com/workflow/50/`. The abbreviations on the input and output stubs (which are not important for understanding the workflow) are as follows: *con* connection, *ctx* context, *odt* Orange data table, *lot* list of Orange data tables, *str* string, *arf* ARFF file, *ins* instances, *lrn* learner, *cla* classifier.

2. *Database Context:* This widget enables a selection of tables and columns that will be used in the next steps of the methodology. The information is carried to the connected widgets through the so-called database context objects. These objects also contain the detected table relationships. In case that the input relational database does not have predefined primary and foreign keys between the tables, the user is given an option for simple table connection search through the names of the attributes.

3. *Dataset Discretization:* The sole task of this widget is to convert continuous attributes to categorical, by discretizing the continuous attributes. Dataset Discretization widget supports three discretization methods: using equal-width intervals, using equal-frequency intervals, and class-aware discretization proposed by [113] that uses MDL and entropy to find the best cut-off points. Dataset Discretization widget can take as input either a single data set or a list of multiple datasets. In the latter case, discretization of all continuous attributes of every dataset is performed.

4. *Wordification:* The wordification widget transforms the relational database to a corpus of documents for the main table. As an input it takes three arguments: the target (main) table, a list of additional tables and a database context, which contains the relations between the tables. The widget first indexes the examples of every table by their primary and foreign keys' values. This step is required for performance optimization of data retrieval operations when searching for connecting instances from different (connected) tables in the word-item concatenation step. Next, recursive document construction for every individual is performed. The algorithm starts on every example of the main table: it creates word-items for its attributes, followed by concatenations of the word-items and results of the recursive search through the connecting tables. When searching along the tree of connected tables, the algorithm stores the results of subtree word concatenations for every instance. Consequently, the algorithm iterates over every subtree only once. The user can run the widget with different parameters: maximal number of n-grams per word, as well as the pruning threshold parameter. The wordification widget produces two outputs: a set of generated word documents and an *arff* table with calculated TF-IDF values for every example of the main table.

5. *Word Cloud:* A set of generated word documents can be displayed using word cloud visualization, enabling improved domain understanding.

6. *Data Mining:* After the wordification step the user can perform various types of analysis, depending on the task at hand (classification, clustering, etc). In the example workflow shown in Figure 6.6, the *arff* output is used as input to build and display a J48 decision tree.

Along with other components from the ILP module, the wordification component can be used to construct diverse RDM workflows. As completed workflows, data, and results can also be made public by the author of the workflow, the platform can serve as an easy-to-access integration platform for various RDM workflows. Each public workflow is assigned a unique URL that can be accessed by anyone to either repeat the experiment, or use the workflow as a template to design another workflow.

6.5 Experiments

This section presents the evaluation of the wordification methodology. After describing the relational databases used in this study, we describe the experiments performed on these datasets and provide a comparison of wordification to other propositionalization techniques. In comparison with the experimental setting described in [98], a larger number of datasets is used, and very favorable results are obtained by using decision tree learner J48, compared to relatively poor results reported in previous work, where the Naive Bayesian classifier assuming feature independence was used. In addition to the J48 tree learner, we also tested the LibSVM learner.

Let us first present the five relational databases used in the experiments: Trains (in two variants), Carcinogenesis, Mutagenensis with 42 and 188 examples, IMDB, and Financial. Table 6.1 lists the characteristics of the datasets. All the datasets can be downloaded from a web page[4], making them easily reusable in other experiments.

Table 6.1: Table properties of the experimental data.

Trains

Table	#rows	#attributes
cars	63	9(10)
trains	20	2

Carcinogenesis

Table	#rows	#attributes
atom	9,064	5
canc	329	2
sbond_1	13,562	4
sbond_2	926	4
sbond_3	12	4
sbond_7	4,134	4

Mutagenesis 42

Table	#rows	#attributes
atoms	1,001	5
bonds	1,066	5
drugs	42	7
ring_atom	1,785	3
ring_strucs	279	3
rings	259	2

Mutagenesis 188

Table	#rows	#attributes
atoms	4,893	5
bonds	5,243	5
drugs	188	7
ring_atom	9,330	3
ring_strucs	1,433	3
rings	1,317	2

IMDB

Table	#rows	#attributes
actors	7,118	4
directors	130	3
directors_genres	1,123	4
movies	166	4
movies_directors	180	3
movies_genres	408	3
roles	7,738	4

Financial

Table	#rows	#attributes
accounts	4,500	4
cards	892	4
clients	5,369	4
disps	5,369	4
districts	77	16
loans	682	7
orders	6,471	6
tkeys	234	2
trans	1,056,320	10

[4]`http://kt.ijs.si/janez_kranjc/ilp_datasets/`

- *Trains:* The well-known East-West Trains challenge is an ILP problem of predicting whether a train is East-bound or West-bound. A train contains a variable number of cars that have different shapes and carry different loads. We have considered two versions of the data for our experiments: the original dataset from the East-West Trains challenge and a modified dataset where every car also has its position as an additional attribute. In both datasets we have considered East-bound trains as positive examples.
- *Carcinogenesis:* The problem addressed by [114] is to predict carcinogenicity of a diverse set of chemical compounds. The dataset was obtained by testing different chemicals on rodents, where each trial would take several years and hundreds of animals. The dataset consists of 329 compounds, of which 182 are carcinogens.
- *Mutagenesis:* In this task the goal is to predict mutagenicity of aromatic and heteroaromatic nitro compounds [115]. Predicting mutagenicity is an important task as it is very relevant to the prediction of carcinogenesis. The compounds from the data are known to be more structurally heterogeneous than in any other ILP dataset of chemical structures. The database contains 230 compounds of which 138 have positive levels of mutagenicity and are labeled as 'active'. Others have class value 'inactive' and are considered to be negative examples. We took the datasets of the original Debnath paper [115], where the data was split into two subsets: a 188 compound dataset and a smaller dataset with 42 compounds.
- *IMDB:* The complete IMDB database is publicly available in the SQL format. This database contains tables of movies, actors, movie genres, directors, and director genres. The database used in our experiments consists only of the movies whose titles and years of production exist on the IMDB's top 250 and bottom 100 chart[5]. The database therefore consists of 166 movies, along with all of their actors, genres and directors. Movies present in the IMDB's top 250 chart were considered as positive examples, while those in the bottom 100 were regarded as negative.
- *Financial:* This is a publicly available dataset, which was artificially constructed as part of the PKDD'99 Discovery Challenge. The classification task addressed is the prediction of successful loans. The dataset consists of 8 tables describing clients of a bank, their accounts, transactions, permanent orders, granted loans and issued credit cards.

Table 6.2 presents the performance of the majority classifier for each of the described datasets. This should serve as a baseline for the classification results reported in the following subsections.

Table 6.2: Majority classifier performance for each dataset.

Domain	CA[%]
Trains	50.00
Mutagenesis 42	69.05
Mutagenesis 188	66.50
IMDB	73.49
Carcinogenesis	55.32
Financial	86.75

[5]As of July 2, 2012.

6.5.1 Evaluation of Feature Construction and Filtering

The experiments, enabling the analysis of the feature generation step of wordification were performed on the original East-West Trains challenge dataset using different parameter settings: using elementary word-items and complex word-items constructed from up to 5-grams of witems.

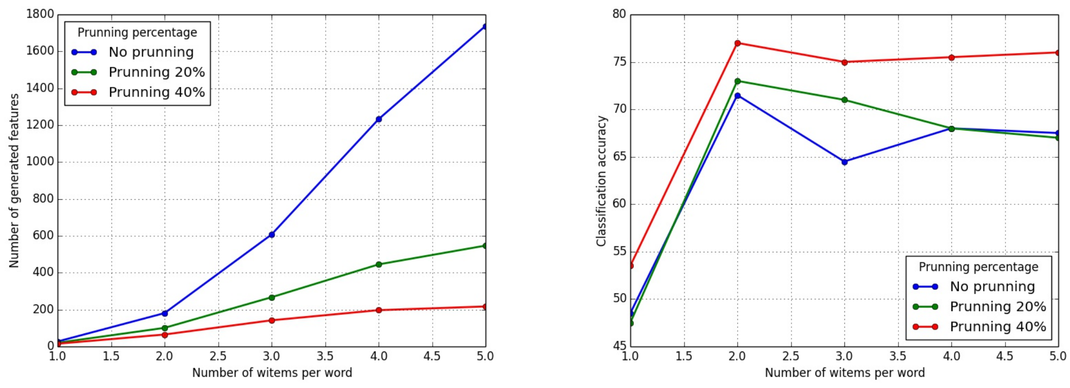


Figure 6.7: Experiments on the wordification propositionalization step. Left: Size of the feature space in correlation with the number witems per word. Right: Classification accuracies in leave-one-out cross-validation (using the J48 decision tree learner with default parameter setting) as a function of the maximum number of n-grams per word.

The left plot in Figure 6.7 shows that the size of the feature space in the non-pruned version of wordification increases exponentially as the maximal number of witems per word increases. Note that wordification also implements a pruning technique where words that occur in less than a predefined percentage of documents are pruned. As shown in Figure 6.7, using higher thresholds for feature filtering drastically reduces the dimensionality of the data, resulting in more efficient learning.

We have also applied different wordification settings in the classification task on the Trains dataset. The classification accuracies using the J48 decision tree of leave-one-out cross-validation for different parameters are shown on the right side of Figure 6.7 (the reason for using leave-one-out instead of the standard 10-fold cross-validation setting is a very small number of instances in the Trains dataset). The results show that using larger n-grams of witems improves the classification accuracies compared to 1-grams of witems, but results in longer run-times of the propositionalization step because of a larger feature space. In this specific domain, pruning also performs favorably in terms of classification accuracy. Additional experiments in Appendix C show (see Table C.2) this observation is only applicable to small domains, while for larger domains with more potential witem combinations this observation does not hold.

6.5.2 Comparative Evaluation of Propositionalization Techniques

This section describes the experiments performed in the evaluation of different propositionalization approaches on binary classification tasks, using the datasets from the five relational domains. Figure 6.8 shows the full experimental workflow (from connecting to a relational database management server to visualizing the experimental results and evaluation). This evaluation workflow is available online in the TextFlows platform, which

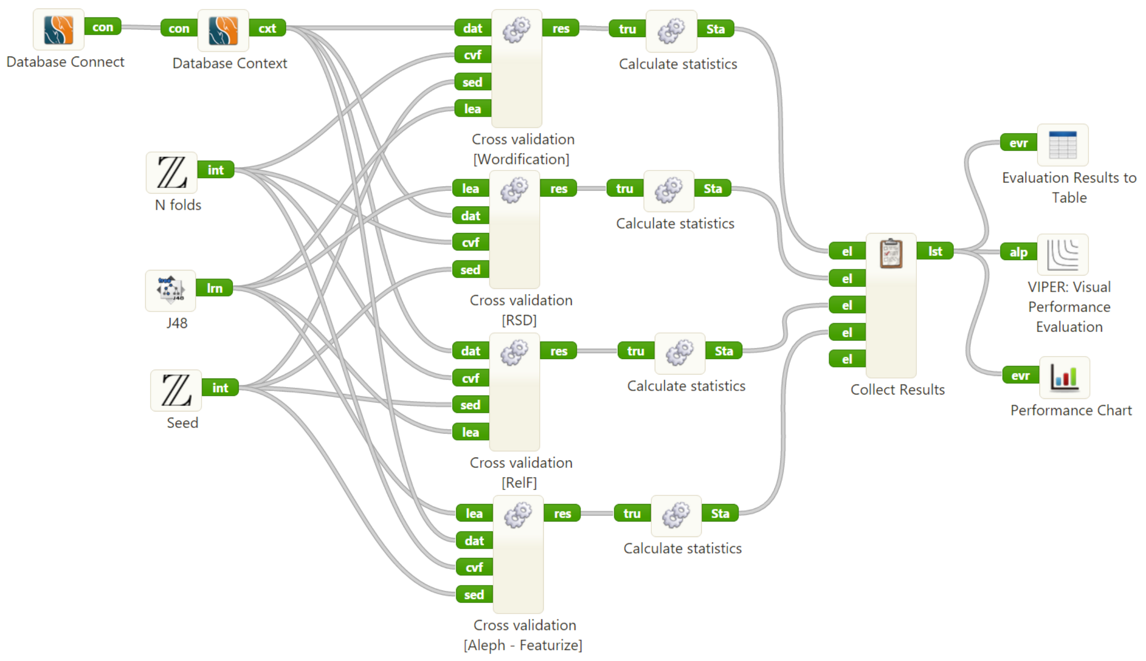


Figure 6.8: Evaluation workflow for evaluating and comparing Wordification, Aleph, RSD, and RelF, implemented in the TextFlows mining platform. The abbreviations on the input and output stubs are as follows: *con* connection, *ctx* context, *dat* full dataset for cross-validation, *cvf* number of cross-validation folds, *sed* random seed, *lrn/lea* learner instance, *res* cross-validation results, *sta* classification statistics, *evr* evaluation results. The workflow is publicly available at `http://textflows.perovsek.com/workflow/69/`.

enables ILP researchers to reuse the developed workflow and its components in future experimentation.

The first step of the evaluation methodology is to read the relational data, stored in an SQL database, using the MySQL package widgets. Data then enters the cross-validation subprocess (Figure 6.9), where the following steps are repeated for each fold (we used 10-fold cross-validation). First, discretization of the training fold of the relational database is performed. We have arbitrarily selected equi-distance discretization with 3 intervals of values to discretize the continuous attributes of the experimental relational datasets, such that none of the techniques was given an advantage. Then a propositionalization technique is applied to the training data and the results are formatted in a way to be used by the Weka algorithms. The J48 decision tree and the LibSVM learners were selected with their default parameter settings to perform binary classification.

The test set is handled as follows. First, the data is discretized to the intervals determined on the training set. Second, the features produced by the given propositionalization approach on the training set are evaluated on the test set to produce a propositional representation of the test data. Note that this process is slightly different for wordification, since the features do not have to be evaluated. We do however need the IDF values calculated on the training set. Finally, these test examples are classified by the classifiers trained on the training data. The results of each step are then collected to be returned at the end by the cross-validation subprocess.

Every propositionalization algorithm was run with its default settings. A non-parallel version of wordification was run using only the elementary words (maximal number of witems per word was set to 1) and without pruning, as none of our datasets required this.

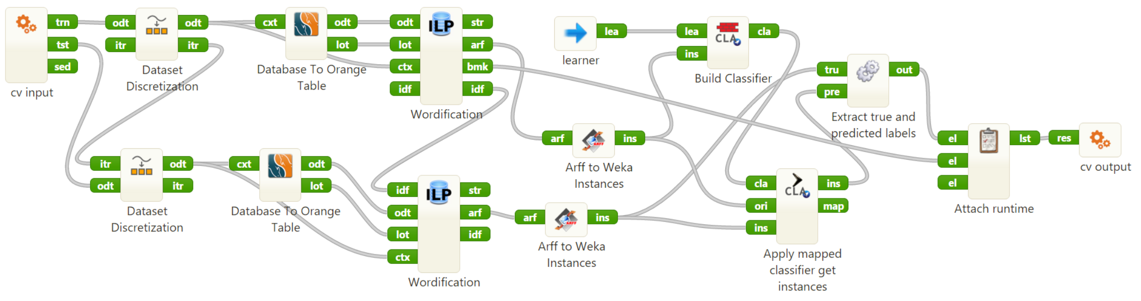


Figure 6.9: The cross-validation subprocess from Figure 6.8. This workflow gets a training set (input *trn*) and a test set (input *tst*) as input and is executed for each fold. There are two Wordification widgets in the workflows: one responsible for constructing the features on the trains set and the other on the test set. The connection between the two widgets is needed for transferring the IDF weights learned on the training set, which are used for feature construction on the test set. The results of each step are collected by the *cv output* widget. Other widget input and output abbreviations are not important for understanding the workflow.

RSD was specified to construct features with a maximum length of a feature body of 8. None of the constructed features were discarded as the minimum example coverage of the algorithm was set to 1.

Aleph was run in the feature construction mode (named AlephFeaturize in the evaluation workflow) with coverage as the evaluation function and maximal clause length of 4. The minimal number of positive examples was set to 1 and the maximal number of false positives to 0.

RelF, the most relevant of the algorithms in the TreeLiker software [76], was run in the default setting as well, but it is not clear from the documentation what exactly are the default parameter values. RelF expects a feature *template* from the user. In this case, we constructed relatively simple templates (enabling features with depth 1), since constructing and selecting more complex templates is out of scope for the analysis in this thesis. It should be noted that templates more finely tuned to a particular domain could yield significantly better results. RelF also supports continuous attributes, but since in our experiments all approaches were given a discretized dataset, this feature could not be exploited.

6.5.3 Results Comparison

The results of the experiments on multiple datasets, presented in Table 6.3, show the classification accuracy and the ROC AUC obtained by the J48 and LibSVM learners (when applied on the data obtained as a result of propositionalization approaches), as well as the run-times needed for propositionalization. The run-time performance for each algorithm was done by measuring the time an algorithm took to propositionalize the full database in each domain. The results show that the wordification methodology achieves scores comparable to the state-of-the-art propositionalization algorithms RSD and RelF, as well as compared to propositionalization performed by using features constructed by Aleph, while the run-time required for transforming the database into its propositional form is much faster.

In terms of classification accuracy obtained by the J48 classifier, wordification performs favorably compared to other propositionalization techniques, except on the Trains dataset (without the car's position attribute) and the Financial dataset. Poor performance on the

Table 6.3: Classifier evaluation on different databases.

Domain	Algorithm	J48		LibSVM		Time[s]
		CA[%]	AUC	CA[%]	AUC	
Trains without position	Wordification	50.00	0.50	50.00	0.50	**0.11**
	RelF	**65.00**	**0.65**	**80.00**	**0.80**	1.04
	RSD	**65.00**	**0.65**	75.00	0.75	0.53
	Aleph - Featurize	60.00	0.60	65.00	0.65	0.40
Trains	Wordification	**95.00**	**0.95**	50.00	0.50	**0.12**
	RelF	75.00	0.75	75.00	0.75	1.06
	RSD	60.00	0.60	**80.00**	**0.80**	0.47
	Aleph - Featurize	55.00	0.55	70.00	0.70	0.38
Mutagenesis 42	Wordification	**97.62**	**0.96**	**78.57**	**0.65**	**0.39**
	RelF	76.19	0.68	76.19	0.62	2.11
	RSD	**97.62**	**0.96**	69.05	0.50	2.63
	Aleph - Featurize	69.05	0.50	69.05	0.50	2.07
Mutagenesis 188	Wordification	68.62	0.55	**81.91**	**0.78**	**1.65**
	RelF	**75.00**	**0.68**	68.62	0.54	7.76
	RSD	68.09	0.54	71.28	0.58	10.10
	Aleph - Featurize	60.11	**0.68**	60.11	0.68	19.27
IMDB	Wordification	**81.93**	**0.75**	73.49	0.50	**1.23**
	RelF	69.88	0.66	73.49	0.50	32.49
	RSD	74.70	0.59	73.49	0.50	4.33
	Aleph - Featurize	73.49	0.50	73.49	0.50	4.96
Carcinogenesis	Wordification	**62.31**	**0.61**	**60.79**	**0.58**	**1.79**
	RelF	60.18	0.59	56.23	0.52	16.44
	RSD	60.49	0.59	56.23	0.52	9.29
	Aleph - Featurize	55.32	0.50	55.32	0.50	104.70
Financial	Wordification	86.75	0.50	**86.75**	0.50	**4.65**
	RelF	**97.85**	**0.92**	86.70	0.50	260.93
	RSD	86.75	0.50	79,06	0.50	533.68
	Aleph - Featurize	86.75	0.50	**86.75**	0.50	525.86

Trains dataset can be explained by examining the J48 tree in the wordified trains dataset with the position attribute, where the J48 classifier puts the cars_position_3 attribute into the root of the decision tree. Because of the absence of this attribute in the first dataset and the usage of only unigram words, the decision tree failed to find a clear distinction between the positive and negative examples (this problem can be solved by using bigrams of witems). Similar results were obtained using the LibSVM classifier where wordification achieved the best results on every dataset except for the two variants of the Trains data.

From the point of view of run-times, wordification is clearly the most efficient system, as it outperforms other techniques on every dataset. The true value of the wordification methodology, its low time-complexity, shows even more drastically on larger datasets, such as Carcinogenesis and Financial datasets, where it achieves comparable classification results in up to 100-times faster manner (compared to RSD or Aleph feature construction).

In order to statistically compare classification accuracies of multiple propositionalization approaches (separately for each of the classifiers) on multiple datasets, we applied the Friedman test [116] using significance level $\alpha = 0.05$ and the corresponding Nemenyi post-hoc test [117]. This approach is used as an alternative to the t-test, which is proven

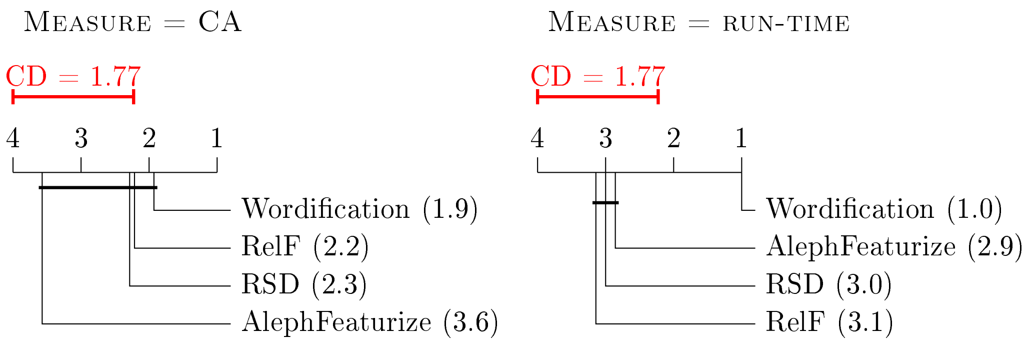


Figure 6.10: Critical distance diagram for the reported classification accuracy (left; not enough evidence to prove that any algorithm performs better) and run-time (right; significant differences for $\alpha = 0.05$) results. The numbers in parentheses are the average ranks.

to be inappropriate for testing multiple algorithms on multiple datasets [118].

The Friedman test ranks the algorithms for each dataset, the best performing algorithm getting the rank of 1, the second best rank 2, etc. In the case of ties, average ranks are assigned. The Friedman test then compares the average ranks of the algorithms. The null-hypothesis states that all the algorithms are equivalent and so their ranks should be equal. If the null-hypothesis is rejected, we can proceed with a post-hoc test, in our case the Nemenyi test. The Nemenyi test is used when we want to compare multiple algorithms to each other. The performance of the algorithms is significantly different if the average ranks differ by at least the *critical distance* (CD), as defined by [118]. This test can be visualized compactly with a critical distance diagram; see Figure 6.10 for classification accuracy (CA) and run-time, when using J48 as the selected classifier (omitting AUC due to similar results obtained as for CA).

The described statistical test was performed using J48 for the three reported measures: classification accuracy, AUC and run-time. The validation yielded the following. For classification accuracy and AUC, there is not enough evidence to prove that any propositionalization algorithm on average performs better than the others (Figure 6.10 left, for significance level $\alpha = 0.05$), even though wordification achieves the best results on 5 out of 7 benchmarks. This is due to the fact that the test takes into account the order of *all* algorithms, not only one versus the others.

We repeated the same statistical analysis for the LibSVM results, where the conclusion ended up the same. For classification accuracy and AUC, there is not enough evidence to prove that any propositionalization algorithm on average performs better than the others, even though wordification also achieves the best results on 5 out of 7 benchmarks.

For run-time, the results are statistically significant in favor of wordification; see the critical distance diagram in the right part of Figure 6.10. The diagram tells us that the wordification approach performs statistically significantly faster than other approaches, under the significance level $\alpha = 0.05$. Other approaches fall within the same critical distance and no statistically significant difference was detected.

As shown in Figure 6.8, the results of the Cross Validation widget (precision, recall, F-score) are connected to the input of the VIPER (Visual Performance Evaluation) widget. VIPER is an alternative evaluation visualization [95], implemented in the TextFlows data mining platform, which displays the results as points in the two-dimensional precision-recall space (for the selected target class). Figure 6.11 presents the VIPER performance visualization, evaluating J48 and LibSVM results after applying wordification, RSD, RelF and

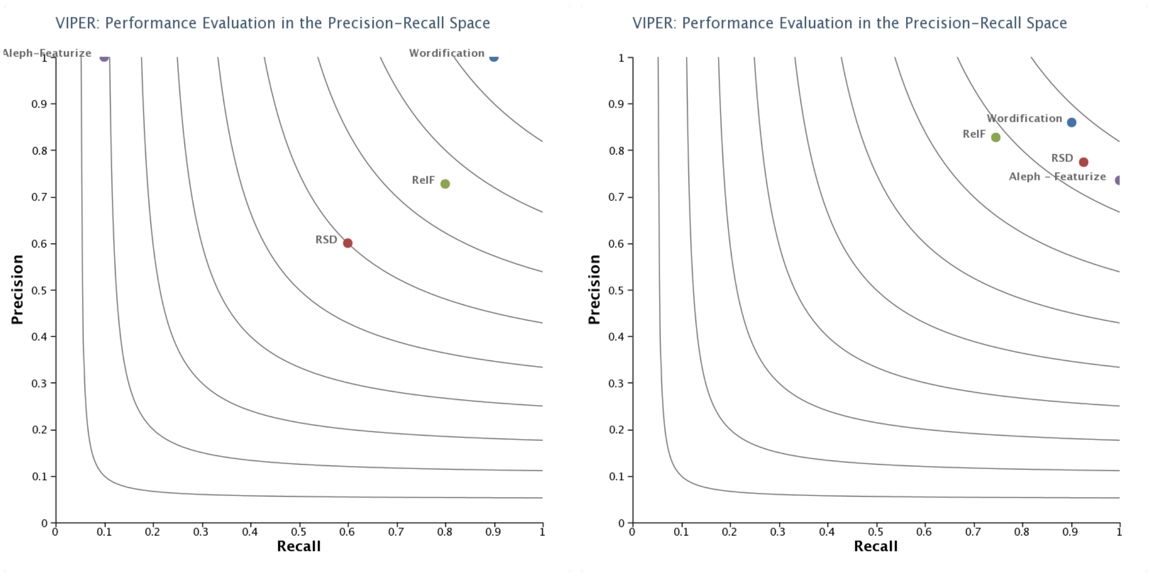


Figure 6.11: The VIPER visualization showing evaluations of the standard J48 algorithm after applying propositionalizaton techniques. In the Trains dataset (left), 'East' was selected as the target class, while in the IMDB dataset (right) positive class was selected as the target.

Aleph feature construction as propositionalizaton techniques. The results are presented in the so-called precision-recall space, where each point represents the result of an algorithm. Points closer to the upper-right corner have higher precision and recall values. F-measure values are presented as isolines (contour lines) in the precision-recall space, which allows a simple comparison of algorithm performances.

From the results shown in Figure 6.11 we can conclude that in terms of precision and recall J48 achieves best results using the wordification propositionalization. Using the wordification methodology, not only a higher percentage of positive examples was retrieved (higher recall score), but also a slightly higher percentage of correctly classified examples of the target class (higher precision score) compared to other propositionalization techniques.

Chapter 7

Summary and Further Work

The work in this thesis facilitates the construction and execution of text mining and natural language processing (NLP) workflows by implementing a web-based text mining platform TextFlows. By unifying functionalities of different knowledge discovery platforms and adding features to facilitate text mining and natural language processing, we have addressed novel scenarios which could not have been considered within the same platform until now, such as comparison of stemmers and POS taggers from different libraries, literature-based discovery in a text mining platform, and relational data mining empowered by a text mining inspired technique.

In the rest of this chapter the achievements and main scientific contributions of the thesis are briefly summarized. We conclude by discussing the directions of further work and planned improvements of the presented platform, developed methodologies, their components and their integrations.

7.1 Summary of Contributions

This thesis presents TextFlows, an open source platform featuring workflow components for text mining and natural language processing. TextFlows provides a common graphical user interface and joins several text mining, visualization and machine learning libraries under a single unifying platform, expanding their usability for researchers, practitioners and non-experts. The platform provides a scalable architecture, which can serve several concurrent users, and offers a framework for simple workflow and dataset sharing.

The usability of the platform was demonstrated on five natural language processing illustrative use cases, showing that components developed for different systems and in different programming languages can be run within a single coherent system and may be represented as a visually constructed workflow available on the web. The ease of sharing the completed workflows and results over the web allows for TextFlows to become a central integration platform for many text mining and natural language processing tasks. We have compared our work to related platforms and shown the differences, benefits, and drawback for using TextFlows with respect to other text mining platforms.

We implemented and presented the TextFlows cross-context literature mining facility, which in combination with the existing term exploration engine CrossBee [17]–[19] supports the expert in advanced document exploration, aimed at facilitating document retrieval, analysis and visualization. The combination of the two systems forms a creativity support tool, helping experts to uncover not yet discovered relations between seemingly unrelated domains from large textual databases. For estimating which terms have a high bisociative potential we implemented a complex methodology which was developed as a pipeline of natural language processing and literature-based discovery components in the TextFlows platform. The visual programming user interface of TextFlows not only enables the user to tailor the methodology steps to his own needs but also allows experiment repeatability and methodology reuse by other users and developers. It was shown that by using a predefined vocabulary we can increase the heuristics' capacities to rank the B-terms at the beginning of the term list. Indeed, by applying this approach in the migraine-magnesium and autism-calcineurin domains we got a higher concentration of B-terms among the best ranked terms. Consequently, the user is presented with a simpler exploration task of finding hidden links that connect seemingly unrelated domains.

Furthermore, the thesis also presents the propositionalization [24], [25] technique called wordification, which aims at constructing a propositional table using simple and easy to understand features. This methodology, implemented as a use case in the TextFlows platform, is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents, where document 'words' are constructed from attribute values by concatenating each table name, attribute name and value (called word-item or witem in this paper) into a single named-entity. As is typical for propositionalization methods, after the wordification step any propositional data mining algorithm can be applied. As shown in the experiments on seven standard ILP datasets, the proposed wordification approach using the J48 and LibSVM classifiers performs favorably (in terms of accuracy and efficiency), compared to state-of-the-art propositionalization algorithms (RSD [75], RelF [76]) as well as compared to propositionalization performed by using features constructed by Aleph [71]. In addition, the proposed approach has the advantage of producing easy to understand hypotheses, using much simpler features than RSD and other systems, which construct complex logical features as conjunctions of first-order literals. It is interesting to observe that in wordification feature simplicity is compensated by the mechanism of feature weighting, inherited from text mining, which successfully countervails the loss of information compared to complex relational features constructed by other propositionalization algorithms.

7.2 Public Availability of the Developed Software

We have released the sources of the TextFlows platform under an open source MIT license, available at `https://github.com/xflows/textflows`. Detailed deployment instructions are provided with the source code, while the user documentation of individual widgets implemented in the platform is available at `http://docs.textflows.org`. A public installation of the platform is available for online use at `http://textflows.org`, while the static version (not updated with new features) containing the exact workflows presented in this thesis is available at `http://textflows.perovsek.com`.

The git repository of the CrossBee's extension for importing datasets and heuristic results from TextFlows is available at `http://source.ijs.si/kt/crossbee/tree/diagonalization`, while the wordification source code is accessible as part of the Python Relational Data Mining package (`https://github.com/xflows/rdm`).

7.3 Assessment of Contributions

In Section 1.3 we described several goals and hypotheses of this thesis. In this section, we reevaluate the proposed TextFlows platform and the presented methodologies with regards to those requirements.

The main hypothesis presented in Section 1.3 is that textual data can be efficiently processed with a system that implements the visual programming paradigm and utilizes computing by means of distributed hardware and software resources. The comparisons in

Section 3.5 show that none of the described text analytics workflow management systems offers a combination of components from different contexts and from different software sources, while still maintaining a high level of simplicity and interconnectivity, as well as providing full-time accessibility and enabling methodology and result sharing. TextFlows is implemented as a cloud-based web application, which helps it overcome various deficiencies of similar text analytics platforms in terms of accessibility and scalability. Moreover, the TextFlows platform is open source and provides several novel features, which are beneficial to the text mining community, such as interoperability between different NLP libraries. However, offering a combination of algorithms from different software libraries through a unified input-output representation also brings higher running times due to additional automated data type conversions. Moreover, running widgets (such as LATINO widgets) in a distributed environment on several machines (through Celery) brings additional overhead due to the requirements of transporting the data over the network. This issue was partially addressed by building serializers/deserializers which compress/decompress the *AnnotatedDocumentCorpus* objects into smaller JSON strings. Furthermore, the ability to inspect every widget's inputs and outputs comes with the requirement that every intermediate result needs to be stored into the database, resulting in a huge number of I/O operations, which brings down the performance of the platform.

We have demonstrated the usability of the platform on five natural language processing use cases featuring implementations and comparisons of methods from several knowledge discovery libraries in a unified workflow environment. By unifying functionalities of different knowledge discovery platforms and adding features to facilitate text mining joined with natural language processing, we have addressed novel scenarios which could not be considered within the same platform until now, such as comparison of stemmers and POS taggers from different libraries, literature-based discovery in a text mining platform, and relational data mining empowered by a text mining inspired technique.

Another hypothesis investigated in this thesis is the usage of advanced text mining approaches for extraction of hidden/uncovered links and patterns across different domains of investigation. We have presented a complex literature-based discovery scenario featuring various text mining components (tokenization, stemming, lemmatization, document visualization, bisociative links discovery) implemented as a workflow in the TextFlows platform. We have shown the extensibility of the platform by implementing several components, which—together with document exploration and visualization features of the CrossBee human-computer interface—make it a powerful, user-friendly text analysis tool for exploratory cross-domain knowledge discovery. An advantage of the TextFlows-CrossBee combination compared to only CrossBee is in the possibility to experiment with different settings, as well as the possibility to extend the methodology with new ideas and then compare or evaluate the developed approaches. Consequently, for end-users who are not computer scientists or data miners the system is a bit harder to use, as such users usually prefer to follow a fixed sequence of predefined methodological steps.

We have also proposed three elementary heuristics based on banded matrices for the bridging term discovery. The evaluation showed that the newly proposed heuristics resulted in very favorable results on the migraine-magnesium domain; however, the performance on the autism-calcineurin dataset suggests that the larger BoW space greatly decreases their B-term discovery capabilities. Moreover, the construction of a banded matrix has a much higher space and time complexities compared to the requirements of other elementary heuristics, making banded matrix-based heuristics much harder to use on larger datasets. Even though we have partly addressed these two issues by proposing an extended methodology, which uses the controlled vocabularies to reduce the search space, further extensive research on more datasets is required. This extended bridging term discovery methodology facilitates the use of controlled vocabularies and enhances the heuristics capabilities to rank the actual B-terms at the top of the ranked term list. Results show that using a predefined controlled vocabulary not only increases the efficiency of the heuristic calculation algorithms, but also tends to improve the relevance of top ranked B-terms. Consequently, the described approach enables the user to perform the exploration task more effectively, potentially leading to faster discoveries of new links between the two domains. With all these features, the TextFlows platform, combined with the CrossBee document exploration interface, has become a publicly available tool, which can support creative discovery of new cross-domain hypotheses.

Finally, we proposed a relational data mining scenario where a text mining inspired approach wordification was evaluated in comparison with existing propositionalization approaches on several relational datasets. The experiments showed that wordification performed favourably but that still suffers from some loss of information, compared to propositionalization methods which construct complex first-order features as a chain of joins of one or more tables related to the target table. We tried to address this feature dependence issue by constructing a-kind-of word 'n-grams' as conjuncts of a predefined number of simple features and using TF or TF-IDF word weighting to capture the importance of these new features. Nevertheless, despite some information loss, wordification has numerous advantages. Due to the simplicity of features, the generated hypotheses are easily interpretable by domain experts. The feature construction step in wordification is very efficient, therefore it can scale well for large relational databases. As wordification constructs each 'document' independently from the other 'documents', a large main table can be divided into smaller batches of examples, which can be propositionalized in parallel. Finally, the wordification approach has the advantage of using techniques developed in the text mining community, such as efficient document clustering or word cloud visualization, which can now be effectively exploited in multi-relational data mining.

7.4 Further Work

In this section we first discuss the directions for further work related to the software platform followed by a discussion on the potential directions for further work on the developed methodologies in cross-context literate mining and relational data mining.

7.4.1 The TextFlows Platform

There are several directions for future work. First, we plan to expand the TextFlows widget repository with additional text preprocessing components (such as chunking, term extraction, syntactic parsing), extend the batch of weighting schemes for generating the BoW models, and include various algorithms for clustering. We also plan to extend the literature-based discovery package to include more widgets for cross-domain bridging term discovery. Furthermore, we will connect the platform to various external tools to better assist the user in the process of exploration and visualization of results, such as TopicCircle [18] for clustering visualization.

Second, by using public data on user workflows, submitted to the public version of the TextFlows platform, we will construct a recommender system based on the data on previously executed workflows that will enable computer-assisted construction of text mining workflows and bring the platform even closer to non-experts in terms of usability.

Third, since current weakness of TextFlows is its inability to deal with very large data, we plan to integrate the stream mining and big data mining features of Apache Spark[1] to

[1]`http://spark.apache.org`

the TextFlows platform. This will allow for performing natural language processing on a larger scale, as well as performing different tasks on streams of data such as website news feeds, or real time data from social networks such as Twitter and Facebook.

Furthermore, we wish to simplify the installation procedures of TextFlows to private servers by providing one-click deployment to services such as Amazon Elastic Compute Cloud (EC2)[2], Google Compute Engine[3] and Microsoft Azure[4].

7.4.2 Cross-Context Literature-Based Discovery

In future work we will also introduce additional user interface options for data visualization and exploration as well as advance the term ranking methodology by adding new sophisticated heuristics which will take into account also the semantic aspects of the data (e.g., by introducing terms representing subject-verb-object triplets). Besides, we will apply the system to new domain pairs to exhibit its generality, investigate the need and possibilities of dealing with domain specific background knowledge, and assist researchers in different disciplines in their explorations which may lead to new scientific discoveries.

Furthermore, introducing more semantic understanding into the ranking could substantially improve the performance. We further plan to improve the set of heuristics, e.g., by introducing a more global view taking into consideration a term's local neighbourhood, and exploring the potential of outlier documents in guiding search.

7.4.3 Relational Data Mining

In our experiments we also considered feature construction using n-grams. However, our preliminary experiments indicate that in larger domains this technique should be coupled with feature selection algorithms, which we plan to address in our further work.

Other advantages of wordification, which will be explored in further work, include the capacity to perform clustering on relational databases; while this can be achieved also with other propositionalization approaches, wordification may successfully exploit document similarity measures and word clouds as easily understandable means of cluster visualization.

The implementation of the entire experimental workflow (from connecting to a relational database management server to visualizing the experimental results and evaluation) in the web-based text mining platform TextFlows is another major contribution, which will enable ILP researchers to reuse the developed software in future experimentation. To the best of our knowledge, this is the only workflow-based implementation of ILP algorithms in a platform accessible through a web browser, enabling simple workflow adaptation to the user's needs. Adding new ILP algorithms to the platform is also possible by exposing an algorithm as a web service. This may significantly contribute to the accessibility and popularity of ILP and RDM methods in the future.

In future work we will also address other problem settings (such as clustering) and use the approach for solving real-life relational problems. Moreover, we plan to use the approach in a more elaborate scenario of mining heterogeneous data sources, involving a mixture of information from databases and text corpora. We will further investigate the strength of n-gram construction and feature weighting, as used in the text mining community, in propositional and relational data mining, as our results indicate that these mechanisms may be successfully used to compensate for the loss of information compared to constructing complex logical features.

[2]`https://aws.amazon.com/ec2/`
[3]`https://cloud.google.com/compute/`
[4]`https://azure.microsoft.com/`

Appendix A

Cross-Domain Literature Mining

A.1 Individual Steps of the CrossBee Methodology in the TextFlows Platform

This section describes the entire underlying pipeline of natural language processing (NLP) and literature-based discovery (LBD) components of the CrossBee methodology (described in Section 5.2.1) implemented as a workflow in the TextFlows platform. While the top-level procedural explanation of the workflow, shown in Figure 5.6, is provided in Sections 5.2.1 and 5.3, the detailed explanations of individual steps of the workflow are described below. Note that the proposal of such workflow was already presented in [19], thus our main contribution is the adapted implementation in the TextFlows platform.

A.1.1 Document Acquisition Workflow

Document acquisition is the first step of the workflow from Figure 5.6. Its goal is to acquire documents of the two domains, label them with domain labels and pack both domains together into the annotated document corpus format. It is composed of several components described below.

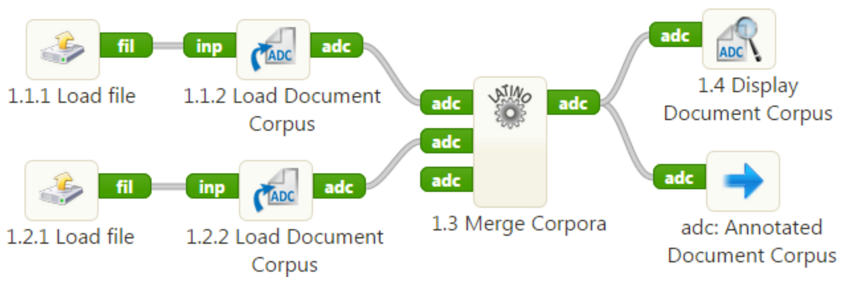

fil
inp
adc
1.1.1 Load file
1.1.2 Load Document Corpus
1.2.1 Load file
1.2.2 Load Document Corpus
adc
1.3 Merge Corpora
1.4 Display Document Corpus
adc: Annotated Document Corpus

Figure A.1: Document acquisition workflow.

The components are responsible for the following tasks:

1.1. load literature A into annotated document corpus data structure

1.1.1. load raw text data from a file (this component could be replaced by loading documents from the web or by acquiring them using web services), where each line contains a document with exactly three tab-separated entries: (a) document identifier, (b) domain acronym, and (c) the document text, 1.1.2. build the annotated document corpus from the raw data, i.e. parse the loaded raw text data into a collection of documents and assign a domain label (e.g., "literature A", "docsA", "migraine") to the documents to enable their identification after merging with literature B,

1.2. load literature B into the annotated document corpus data structure (individual components are aligned with the components 1.1),
1.3. merge the two literatures into a single annotated document corpus structure,
1.4. optional check of document acquisition by visual inspection of the created corpus.

The document acquisition workflow is shown in Figure A.1. The output is the annotated document corpus consisting of the acquired documents labeled with domain labels.

A.1.2 Text Preprocessing Workflow

The document acquisition step is followed by the text preprocessing step, which is itself a workflow implemented as shown in Figure A.2. The main components here are tokenization, stopwords labeling and token stemming or lemmatization. The output of this step is structurally equal to the input; however every document in the annotated document corpus now contains additional information about tokens, stopwords and lemmas.

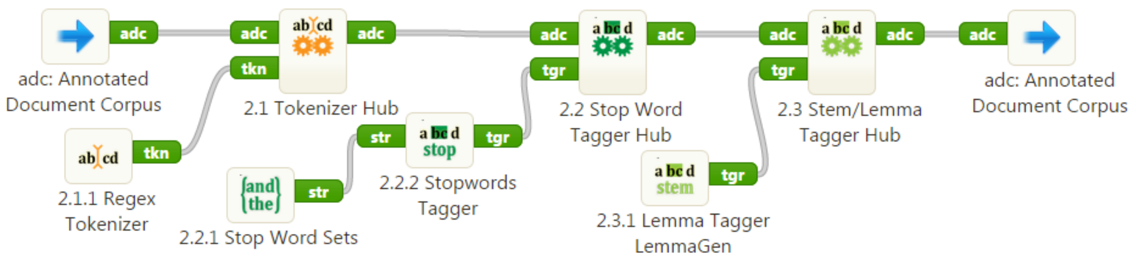


Figure A.2: Document preprocessing workflow.

The individual components perform the following tasks:

2.1. split documents to tokens (the basic units for further text processing),
   2.1.1. create tokenizer object (simple tokenizer based on regular expressions),
2.2. tag stopword tokens by using a stopword tagger (component 2.2.2),
   2.2.1. load standard English stopwords,
   2.2.2. define the stopword tagger using the standard English stopwords only (the detected stopwords are used in candidate B-term extraction step),
2.3. lemmatize tokens by applying the LemmaGen lemmatizer[1] [91],
   2.3.1. create an instance of LemmaGen lemmatizer.

A.1.3 Heuristics Specification Workflow

While the heuristics specification step is the core part of our methodology, this step only specifies which heuristics are selected and how these heuristics should be combined into the ensemble heuristic. The actual calculation is performed later in the heuristic term score calculation step.

[1]LemmaGen is an open source lemmatizer with 15 prebuilt European lexicons. Its source code and documentation is publicly available at `http://lemmatise.ijs.si/`

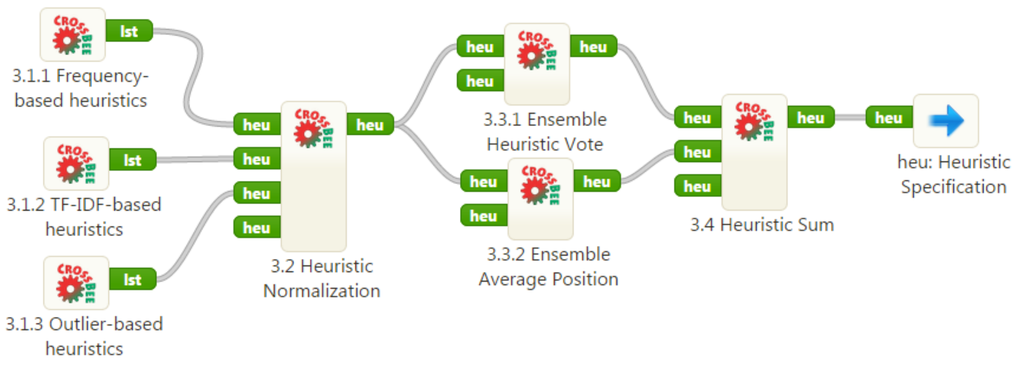

lst
3.1.1 Frequency-based heuristics
lst
3.1.2 TF-IDF-based heuristics
lst
3.1.3 Outlier-based heuristics
heu
heu
heu
heu
heu
3.2 Heuristic Normalization
heu
heu
heu
3.3.1 Ensemble Heuristic Vote
heu
heu
heu
3.3.2 Ensemble Average Position
heu
heu
heu
heu
3.4 Heuristic Sum
heu
heu: Heuristic Specification

Figure A.3: Heuristic specification.

Workflow specification displayed in Figure A.3 is the outcome of our research about the base term heuristics and their combination into the ensemble heuristic presented in Section 5.2.2. Which heuristics to use and how to combine them is based on the experiments on the real data that we performed as a part of the research presented in this thesis—these experiments are presented in more detail in [19]. The findings resulted in the setting shown in Figure A.3, which is a good choice when applied on new data. Nevertheless, the setting and the choice of the base heuristics is fully customizable and can be freely configured to better suit the needs of new applications.

The output of this procedure is a specification of a complex ensemble heuristic, which computes the term bisociation scores. The components in the heuristic specification perform the following tasks:

3.1. define base heuristics (see Section 5.2.2 for details about the base heuristics selection),
   3.1.1. define TF-IDF based heuristic *tfidfDomnSum*,
   3.1.2. define term frequency based heuristic *freqRatio*,
   3.1.3. define outlier based heuristics *outFreqRelRF, outFreqRelSVM, outFreqRelCS, outFreqRelSum*
3.2. for every inputted heuristic defines a new heuristic that normalizes the scores to the range [0,1) and outputs a list of new heuristic specifications,
3.3. combine the six heuristics into a single ensemble heuristic
   3.3.1. define an ensemble voting heuristic that includes votes of the six heuristics (ensemble voting score, see Equation 5.1),
   3.3.2. define a calculated heuristic that calculates normalized sum of position scores of the six heuristics (ensemble position score, see Equation 5.2),
3.4. define the final ensemble heuristic by summing the ensemble voting heuristics, which results in the number of terms heuristics' votes in the range from 0 to 6 (integer value), and the calculated normalized sum of heuristics scores in the range from 0 to less than 1 (final ensemble score, see Equation 5.3).

A.1.4 Candidate B-term Extraction Workflow

Another core step of the workflow is candidate B-term extraction, shown in Figure A.4. Although it contains only one component, it has a very important and complex goal of transforming the inputted annotated document corpus into the BoW model in order to represent documents in the form of feature vectors of term occurrences in the documents

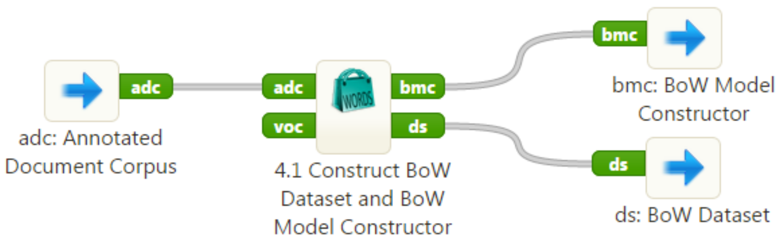

adc
adc: Annotated Document Corpus
adc
voc
bmc
ds
4.1 Construct BoW Dataset and BoW Model Constructor
bmc
bmc: BoW Model Constructor
ds
ds: BoW Dataset

Figure A.4: Candidate B-term extraction.

(for the purpose of visualization of documents and the need of highlighting and emphasizing of specific terms). Another task of this step is to capture the exact parsing procedure, which is needed in order to perform various computations which are performed in the advanced heuristic term scores calculation step. The outputted *BowModelContructor* object also contains the vocabulary of all terms.

A.1.5 Heuristic Term Score Calculation Workflow

Figure A.5 shows a structurally simple methodological step of heuristic term score calculation that contains only one component. The inputs to the procedure are the annotated document corpus, the *BoWModelContructor* and the heuristics specification. Based on the information present in the *BoWModelContructor*, the algorithm calculates various frequency and TF-IDF document features vectors, which are used to calculate the specified heuristics scores for all the terms. The calculation results in the same heuristic structure as defined in the heuristic specification step, however the ensemble heuristic at the top level, as well as all elementary heuristics, now contain their calculated scores of the terms. The scores of the top-level heuristic are intended to represent terms' bisociation scores and are typically used as a basis for the final term ranking.

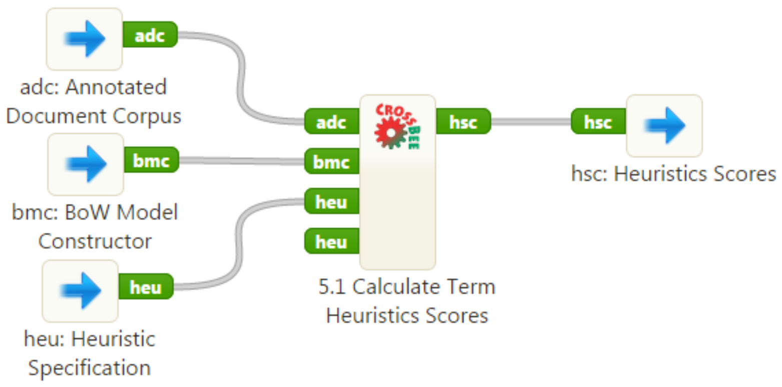

adc
adc: Annotated Document Corpus
bmc
bmc: BoW Model Constructor
heu
heu: Heuristic Specification
adc
bmc
heu
heu
hsc
5.1 Calculate Term Heuristics Scores
hsc
hsc: Heuristics Scores

Figure A.5: Heuristic term score calculation.

A.1.6 B-term Visualization and Exploration Workflow

Step 6 of the methodology implements a workflow shown in Figure A.6. It enables visualization and exploration of the ranked list of B-terms. There are four inputs to this step. The first and the most important are the ensemble heuristic scores of the extracted candidate B-terms. Inputs *Annotated Document Corpus* and *BoW Dataset* are used by

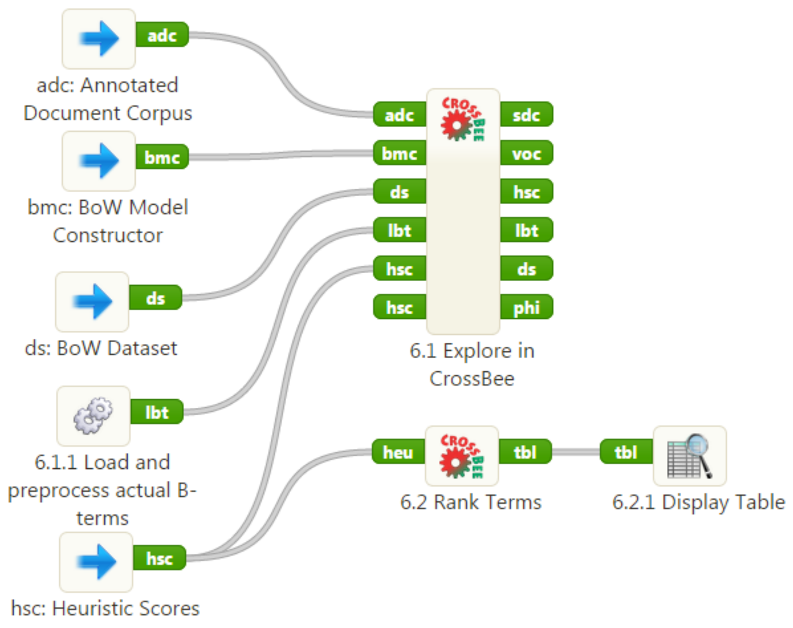


Figure A.6: B-term visualization and exploration.

the online application for cross-context bisociation exploration CrossBee, which needs the exact information about term extraction from documents to be able to align the terms back with the original documents in order to visualize them; while the *BoW Model Constructor* provides the constructed vocabulary. The goals of the created components are the following:

6.1. explore the final results in a web application CrossBee, which was designed specifically for the purpose of bisociativity exploration (expressed either through terms or through documents),

  6.1.1. optional expert specified B-terms may be provided to CrossBee in order to emphasize them in the text and to deliver a feedback about the bisociative quality of the provided ranking. If available, these terms are loaded and preprocessed using the same preprocessing techniques as described in the document preprocessing step,

6.2. rank the terms

  6.2.1. display the ranked terms in the form of a table along with their respective scores.

A.1.7 Methodology Evaluation Workflow

Step 7 of the overall methodology is the methodology evaluation step, implemented as a workflow shown in Figure A.7. There are three inputs to the process: the heuristic scores of one or more evaluated heuristics (which presents the result of all the preceding methodological steps), the *BowModelContructor* (which contains the corpus vocabulary) and additional information about the actual B-terms (required in order to assess any kind of quality measures). Note that, in order not to overflow the overall methodology workflow of Figure 5.6 with additional information, the list of actual bridging terms was not shown as an additional step of the methodology. Instead, it is implemented as a separate subprocess in the methodology evaluation workflow, which is responsible for loading and preprocessing the actual B-terms.

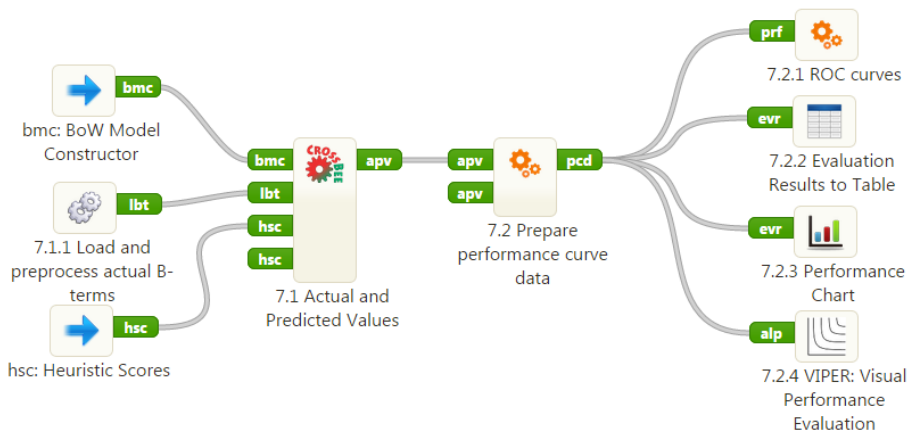

bmc
bmc: BoW Model Constructor
lbt
7.1.1 Load and preprocess actual B-terms
hsc
hsc: Heuristic Scores
bmc
lbt
hsc
hsc
apv
7.1 Actual and Predicted Values
apv
apv
pcd
7.2 Prepare performance curve data
prf
7.2.1 ROC curves
evr
7.2.2 Evaluation Results to Table
evr
7.2.3 Performance Chart
alp
7.2.4 VIPER: Visual Performance Evaluation

Figure A.7: Methodology evaluation.

The components of the methodology evaluation workflow perform the following tasks:

7.1. prepare pairs of actual and predicted values, which are used to calculate different information retrieval measures in step 7.2,

 7.1.1. if available, load the actual (expert identified) B-terms, which present the gold standard terms used to evaluate the quality of the methodology and preprocess them using same techniques as in document preprocessing step,

7.2. calculate different measures, such as precision, recall, and the F_1-measure, ROC curves and the AUC (Area Under Curve) values,

 7.2.1. display ROC curves graphically,
 7.2.2. compare information retrieval measures in the form of a table,
 7.2.3. compare information retrieval measures in the form of a bar chart,
 7.2.4. display and compare the F_1-scores in the advanced VIPER performance evaluation chart [95] component.

The methodology evaluation functionality presented in this section is not part of the actual workflow for cross-domain knowledge discovery; however, it is indispensable when developing a new approach. Description of this step concludes the section presenting the key parts of the methodology.

A.2 Heuristic Scores on the Migraine-Magnesium Dataset

In Section 5.5.3 we demonstrated that by using a predefined controlled vocabulary we can increase the heuristics' capabilities to rank the B-terms at the beginning of the term list on the migraine-magnesium domain, which was described in Section 5.2.4. Tables 5.7 and 5.8 included only the newly presented heuristics based on banded-matrices and the ensemble heuristic and base heuristics included in the ensemble (as identified by Juršič et al.). In this section we show the full list of base heuristics and their scores. The comparison of the heuristics' capabilities to rank the B-terms at the beginning of the term list in the migraine-magnesium domain, presented in Tables A.1 and A.2 below, confirms the advantage of using the controlled vocabulary. By inspecting the number of B-terms found by the heuristics in the ranked first n terms, we notice that using the controlled vocabulary in the migraine-magnesium domain resulted in a much higher concentration of Swanson's B-terms among the best ranked terms for all base heuristics as well as the ensemble.

Table A.1: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the migraine-magnesium dataset.

Heuristic Name	Number of B-terms among top n ranked terms								
	5	10	20	50	100	200	500	1,000	2,000
outFreqRelRf	0.24	0.48	0.95	1	3.72	7.92	15.8	30.33	43
heuristic3	0	0	0	1	3	9	14.92	28.48	43
outFreqRelSum	0	0.36	1	1.64	3.54	7	15	27	43
outFreqRelSvm	0.12	0.24	0.48	1	1.63	5.92	14.69	29	43
heuristic2	0	0	0	1	6	7.91	12.65	29.62	43
outFreqRf	0	0	0	0	0	3.91	16.03	27.86	43
freqDomnRatioMin	0.14	0.28	0.57	1.42	2.83	5.66	14	28	43
heuristic1	0	0	0	1	2	4.48	16.5	26.58	43
outFreqSum	0	0	0	0	0	2	14.28	27.11	43
outFreqSvm	0	0	0	0	0.32	3	14.09	26.14	43
outFreqRelCs	0	0	1	1	2.69	5.07	11	27	43
freqDomnProdRel	0	0	0	0	0	6	14	27	43
outFreqCs	0	0	0	0	0	1.43	15.62	24.67	43
tfidfDomnSum	0	0	0	0	1	2	11	26.14	43
tfidfAvg	0	0	1	1.79	3.11	5.75	11.84	20.9	43
freqDomnProd	0	0	0	0	1	3	13.17	27.16	43
tfidfDomnProd	0	0	0	0	1	3	13.5	27	43
freqRatio	0	0	1	1	4	5	11.65	23.09	43
appearInAllDomains	0.11	0.23	0.46	1.15	2.3	4.6	11.49	22.98	43
tfidfSum	0	0	0	0	0	1	9	25.36	43
freqTerm	0	0	0	0	0	1	8.91	25.49	43
freqDoc	0	0	0	0	0	1	8.03	24.79	43
ensemble	1	1	1	5	6	9	18.57	28	43

Table A.2: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the migraine-magnesium dataset using a controlled vocabulary.

Heuristic Name	Number of B-terms among top n ranked terms								
	5	10	20	50	100	200	500	1,000	2,000
heuristic2	2	5	5.43	12	15.5	25	26	26	26
heuristic3	0	2	6	7	14	22.68	26	26	26
heuristic1	1	2	2.25	9	13.63	20.15	26	26	26
freqDomnRatioMin	0.59	1.18	2.37	5.92	13.25	20	26	26	26
outFreqSum	0	2	3	5.45	15.05	16.96	26	26	26
freqDomnProdRel	0	1	2	5.67	9	20	26	26	26
outFreqRf	1	1.67	3.11	7.93	11.88	15.88	26	26	26
outFreqSvm	1	1	2.5	5.32	11.79	16.79	26	26	26
outFreqCs	0	0	2.45	5.6	10.22	17.06	26	26	26
tfidfDomnSum	0	1	1	4	10	19	26	26	26
freqDomnProd	0	1	1	4	9	19	26	26	26
tfidfDomnProd	0	1	1	4	9	19	26	26	26
outFreqRelRf	0.56	1.11	2	5	8	15	26	26	26
freqDoc	0	0	1	2.5	7.82	17.1	26	26	26
tfidfSum	0	0	1	2.25	7.5	17.35	26	26	26
freqTerm	0	0	1	2.25	7.56	17.43	26	26	26
appearInAllDomains	0.39	0.78	1.56	3.9	7.81	15.62	26	26	26
outFreqRelSum	0.42	0.83	1.2	4	9	14	26	26	26
tfidfAvg	0	1.42	2.47	5.63	7	13	26	26	26
outFreqRelSvm	0.45	0.91	1.82	3.25	9	15	26	26	26
outFreqRelCs	0.31	0.63	1	5	7.06	14	26	26	26
freqRatio	0	1	1	2	5.96	14.56	26	26	26
ensemble	1	3	4	9	13	19	26	26	26

A.3 Heuristic Scores on the Autism-Calcineurin Dataset

In Section 5.5.3 we performed experiments using a predefined controlled vocabulary for cross-domain knowledge discovery on the autism-calcineurin dataset; we have shown only results for the ensemble heuristic and the base heuristics included in the ensemble (see Tables 5.10 and 5.11). In this section we show the experimental results for the full list of base heuristics. The comparison of all heuristics' capabilities to rank the B-terms at the beginning of the term list on the autism-calcineurin dataset, presented in Tables A.3 and A.4 below, confirms the advantage of using the controlled vocabulary on other heuristics as well.

Table A.3: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the autism-calcineurin dataset.

Heuristic Name	Number of B-terms among top n ranked terms										
	5	10	20	50	100	200	500	1,000	2,000	5,000	all
freqRatio	1	1	1	1	1	1	1	3	5	8.99	13
tfidfSum	0	0	0	0	1	1	1	2	4	5	13
tfidfDomnProd	0	0	0	0	0	0	1	1	4	6	13
tfidfDomnSum	0	0	0	0	1	1	2	2	4	5	13
freqTerm	0	0	0	0	1	1	1	2	3	5	13
freqDomnProd	0	0	0	0	0	0	0	1	2	6	13
freqDomnProdRel	0	0	0	0	0	0	0	1	1	6	13
outFreqSum	0	0	0	0	0	1	1	3	5	7	13
outFreqCs	0	0	0	0	0	1	1	1	4	7.17	13
freqDoc	0	0	0	0	0	1	1	2	2	5	13
outFreqRf	0	0	0	0	0	1	1	2.07	4.67	7.39	13
outFreqSvm	0	0	0	0	1	1	1.46	4	4.67	5.44	13
tfidfAvg	1	1	1	1	1	1	1	1	3	6	13
heuristic3	0	0	0	0	0	0	1	4	4	4	13
heuristic1	0	0	0	0	0	0	0	1	4	4.02	13
outFreqRelCs	0	0	0	0	0	0	0	0	0	1.49	13
outFreqRelSum	0	0	0	0	0	0	0	1.64	2	2	13
outFreqRelRf	0	0	0	0	0.01	0.58	1	1	1	2	13
outFreqRelSvm	0	0	0	0	0	0	0	0	1.25	2	13
heuristic2	0	0	0	0	0	0	0	0	0	1	13
appearInAllDomains	0	0	0	0.01	0.02	0.03	0.08	0.17	0.33	0.83	13
freqDomnRatioMin	0	0	0	0	0	0	0	0	0	0	13
ensemble	0	1	1	1	1	2	3	4	6	8	13

Table A.4: Comparison of base and ensemble heuristics capacity to rank the B-terms at the very beginning of the term list for the autism-calcineurin dataset using a controlled vocabulary.

Heuristic Name	Number of B-terms among top n ranked terms										
	5	10	20	50	100	200	500	1,000	2,000	5,000	all
outFreqSvm	0	0	0	0	2	4	4.81	7	8.94	9	9
outFreqSum	0	0	0	0	0	4	5.85	7	8	9	9
tfidfDomnProd	0	0	0	0	0	3	4	7	9	9	9
freqDomnProd	0	0	0	0	1	3	4	7	9	9	9
freqRatio	1	1	1	1	2	3	3.6	6.01	9	9	9
freqDomnProdRel	0	0	0	0	0	1	4	7	9	9	9
outFreqCs	0	0	0	0	0	2	6.59	7	7.82	9	9
tfidfSum	0	1	1	1	1	2	3	7	9	9	9
tfidfDomnSum	0	1	1	1	1	2	3	7	9	9	9
freqTerm	0	1	1	1	1	2	3	6.21	9	9	9
outFreqRf	0	0	0	0	0	3	4.25	6.56	8.01	9	9
freqDoc	0	1	1	1	1	2	3	6	8	9	9
outFreqRelSvm	0	0	1	1	1	1	2	4	9	9	9
heuristic1	0	1	1	1	1	2	2	5	7	9	9
tfidfAvg	1	1	1	1	1	2	2	4	7	9	9
outFreqRelCs	0	0	0	0	0	0	2	3	7	9	9
outFreqRelSum	0	0	0	1	1	1	1	3	7	9	9
heuristic3	1	1	1	1	1	1	1	4	6	9	9
outFreqRelRf	0	0	0	0	0	1	1	3	8	9	9
heuristic2	0	0	0	1	1	1	2	2	5.76	9	9
appearInAllDomains	0.01	0.03	0.06	0.14	0.28	0.55	1.38	2.76	5.52	9	9
freqDomnRatioMin	0	0	0	0	0	0	1	2	6	9	9
ensemble	1	1	1	2	2	2	4	6	8	9	9

Appendix B

Bisociative Morals Ideation

Although not primarily designed for this task, our banded matrix-based B-term ranking methodology, described in Section 5.4, can be used for creating pairs of sentences from different domains, which combine into surprising, funny or even insightful pieces of text when put together and considered as a whole. Bridging terms, appearing in both sentences, detected by our methodology function as a kind of glue, contributing to the coherency and increasing the potential for combinations to be meaningful.

To illustrate the potential of the proposed approach for narrative ideation, we chose two domains: What-if sentences and Aesop's fables' morals. The first domain consists of 94 What-if sentences, retrieved from Twitter (using hash tag #whatif) and the UKWaC British English web corpus. For instance, here is example of such a sentence: *"What if Google someday went down and we couldn't Google what happened to Google?"* The Aesop's fables[1] morals dataset is a collection of 277 fable morals. We created this dataset by crawling the Aesop's fables online collection.

Each what-if sentence as well as each Aesop's fables' moral was treated as a separate document. The documents from both domains were preprocessed using standard text mining techniques, described in Section 4.1. This resulted in 383 distinct terms from all the obtained documents. We applied our methodology to find terms with the highest b-term potential. For simplicity, the terms were sorted using the scores of Heuristic 3 (presented in Section 5.4.5), which we consider the most complete among the presented heuristics. The five terms with the highest b-term potentials were used to create all pairs of sentences sharing the selected bridging term (with the first sentence being from the what-if domain and the second from the Aesop's fables domain). The 15 highest scoring b-term based concatenated pairs of sentences which resulted are shown in Table B.1. These results show–subjectively–that using the terms with the highest b-term potential resulted in several meaningful, creative combinations of sentences. Moreover, it is clear that a large proportion of the sentence pairs in Table B.1 have meaningful relations which could form the basis of artefacts such as poems or stories. We can argue that it would be quite a laborious task to find similarly valuable combinations from all possible pairs (143) of sentences from the given domains without guidance provided by bridging terms. We plan to use crowd-sourcing to test the hypothesis that our approach can reliably produce such valuable combinations.

The experimental evidence indicates that the presented methodology has the potential for supporting the users in the task of bisociative, cross-domain narrative ideation. Banded matrices help us to discover the structures which reveal the nature of relations between terms and documents. We have shown that the approach can be used to construct creative

[1]Aesop was a Greek fabulist, known as author of numerous fables; these are characterized by animals, which solve problems and have human characteristics. See `http://www.aesopfables.com/`

Table B.1: Results of our methodology: combinations of what-if sentences with Aesop's fable morals. In the brackets are the fable titles, which were not part of the document's contents, but are given here as an additional piece of information. The bridging terms are shown in bold.

What if **life** is one big dream, and when we die, we wake up. Evil tendencies are shown in early life. (The Man, the Boy and the Donkey)
What if we woke up, as a baby, and our whole life had been a dream? Evil tendencies are shown in early **life**. (The Man, the Boy and the Donkey)
What if, like Bhutan, we gauged our **life**'s success by how happy we are, not by how big the house is, the number. Evil tendencies are shown in early **life**. (The Man, the Boy and the Donkey)
What if someone you **love** dearly gave you a surprise bday party and when you arrived every1 was exchanging gifts but not nary a 1 was for you! Misery **loves** company. (The Fox Who Had Lost His Tail)
What if someone you **love** dearly gave you a surprise bday party and when you arrived every1 was exchanging gifts but not nary a 1 was for you! Even the wildest can be tamed by **love**. (The Lion in Love)
What if there are other beings in the same room as us but we can't **see** them and they can't see us. Not everything you **see** is what it appears to be. (The Dancing Monkeys)
What if there are other beings in the same room as us but we can't **see** them and they can't see us. Gossips are to be **seen** and not heard. (The Eagle the Cat and the Wild Sow)
What if we don't **see** tomorrow and everything you said today couldn't be undone. Would you be proud? Not everything you **see** is what it appears to be. (The Dancing Monkeys)
What if we don't **see** tomorrow and everything you said today couldn't be undone. Would you be proud? Gossips are to be **seen** and not heard. (The Eagle the Cat and the Wild Sow)
What if eyeballs had butts but we couldn't **see** them because they're hidden in our skulls? Not everything you **see** is what it appears to be. (The Dancing Monkeys)
What if eyeballs had butts but we couldn't **see** them because they're hidden in our skulls? Gossips are to be **seen** and not heard. (The Eagle the Cat and the Wild Sow)
What if humans could actually breathe in space? And the government says we can't so we don't try to **escape**? Nothing **escapes** the master's eye. (The Stag in the Ox-Stall)
What if humans could actually breathe in space? And the government says we can't so we don't try to **escape**? We had better bear our troubles bravely than try to **escape** them. (The Kings Son and the Painted Lion)
What if you simply stopped doing whatever it is that isn't a part of your **success**? The best intentions will not always ensure **success**. (The Monkeys and Their Mother)
What if, like Bhutan, we gauged our life's **success** by how happy we are, not by how big the house is, the number. The best intentions will not always ensure **success**. (The Monkeys and Their Mother)

combinations of sentences from different domains, coupled with bridging terms with the highest b-term potential. The results confirm the potential of the proposed approach to identify meaningful bridging concepts in the intersection of texts from different domains.

In further work, we will upgrade the methodology to combine not only pairs of sentences from two domains, but also to compose longer chains of sentences, resulting in narrative ideation. Another line of research will address more subtle connections between sentences. For instance, there is a semantic connection between *baby* and *early life* in *"What if we woke up, as a baby, and our whole life had been a dream? Evil tendencies are shown in early life."* However, this connection arises purely by coincidence: it was not detected by the system. An important next step will be to crowd-source opinions about how reliable the process is at producing sentence pairs (and larger constructs) which can be meaningfully interpreted in such a way that intelligence and possibly creativity are projected onto the software producing them. After the analysis of the results and further refinement of the techniques, we plan to embed fictional ideation processes and idea expansion via bisociation into software for generating artefacts of cultural value such as poems and stories. We hope to show that the kinds of cross-context link discovery methods presented here can be used generically in Computational Creativity projects across domains.

Appendix C

Evaluation of Wordification Feature Construction

In addition to the TF-IDF weighting scheme, the implementation of wordification (described in detail in Section 6.4) includes also the term frequency (TF) and the binary (0/1) weighting schemes. A comparison of the three schemes can be found in Table C.1. The results show that different weighting schemes, used in our experiments, do not perform significantly differently on the classification tasks.

Table C.1: Evaluation of different feature weighting techniques.

Domain	Weighting	J48-Accuracy[%]	J48-AUC
Trains	TF-IDF	50.00	0.50
without position	TF	**85.00**	**0.85**
	Binary	35.00	0.35
Trains	TF-IDF	**95.00**	**0.95**
	TF	80.00	0.80
	Binary	70.00	0.70
Mutagenesis 42	TF-IDF	97.62	0.96
	TF	97.62	0.96
	Binary	97.62	0.96
Mutagenesis 188	TF-IDF	**68.62**	**0.55**
	TF	68.09	0.54
	Binary	**68.62**	**0.55**
IMDB	TF-IDF	81.93	0.75
	TF	81.93	0.75
	Binary	81.93	0.75
Carcionogenesis	TF-IDF	62.31	0.61
	TF	62.61	0.61
	Binary	**62.92**	**0.62**
Financial	TF-IDF	86.75	0.50
	TF	86.75	0.50
	Binary	86.75	0.50

In Section 6.5.1 we described how we have applied different wordification settings in the classification task on the Trains dataset. The classification accuracies using the J48 decision tree of leave-one-out cross-validation for different parameters showed that using larger n-grams of witems improved the classification accuracies compared to 1-grams of witems. The results of additional experiments on more domains (presented in Table C.2) reveal that this observation is only applicable to small domains, while for larger domains with more potential witem combinations this observation does not always hold.

Table C.2: Evaluation of different number of witems.

Domain	k	J48-Accuracy[%]	J48-AUC	Time[s]
Trains	1	50.00	0.50	0.12
without position	2	75.00	0.75	0.15
	3	75.00	0.75	0.20
Trains	1	95.00	0.95	0.12
	2	75.00	0.75	0.16
	3	70.00	0.70	0.22
Mutagenesis 42	1	97.62	0.96	0.65
	2	97.62	0.96	0.83
	3	92.86	0.88	0.88
Mutagenesis 188	1	68.62	0.55	1.25
	2	68.62	0.55	2.26
	3	66.49	0.50	2.68
IMDB	1	73.49	0.50	0.16
	2	73.49	0.50	0.20
	3	73.49	0.50	0.25
Carcionogenesis	1	56.84	0.56	5.31
	2	51.67	0.51	6.65
	3	52.58	0.51	7.04
Financial	1	86.75	0.50	4.11
	2	86.75	0.50	4.24
	3	86.75	0.50	4.38

Appendix D

Illustrative Examples of the Wordification Methodology

This appendix illustrates the utility of wordification in a descriptive induction setting of learning association rules from two real-life domains, using data from a subset of the IMDB movies database and from a database of traffic accidents. Tables D.1 and D.2 list the characteristics of both databases.

Table D.1: Table properties of the two experimental datasets.

IMDB

Table	#rows	#attributes
movies	166	4
roles	7,738	2
actors	7,118	4
movie_genres	408	2
movie_directors	180	2
directors	130	3
director_genres	243	3

Accidents

Table	#rows	#attributes
accident	102,756	10
person	201,534	10

Table D.2: Document properties after applying the wordification methodology.

Domain	Individual	#examples	#words	#words after filtering
IMDB	movie	166	7,453	3,234
Accidents	accident	102,759	186	79

The preprocessing procedure was performed on the two databases as follows. First, the wordification step was applied. As shown in Figure D.1, we used TextFlows to read the relational data from the MySQL database, discretize continuous attributes and apply the propositionalization step. Due to lack of support for association rule learning in the TextFlows platform, the results of the wordification feature construction step were saved as an ARFF file and imported into RapidMiner [119]. Using RapidMiner we first removed irrelevant features (which have the same value across all the examples), which resulted in the reduction of the features to less than half of the original (see Table D.2). In order to prepare the data for association rule mining, we also binarized the data: after experimenting with different TF-IDF thresholds, features with a higher TF-IDF weight than 0.06 were assigned the value *true* and *false* otherwise.

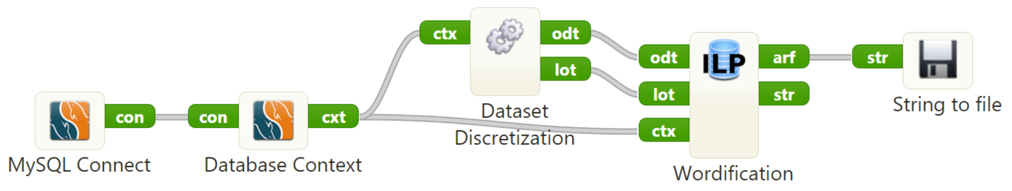


Figure D.1: TextFlows wordification workflow used for feature construction before applying association rule learning. This workflow is publicly available at `http://textflows.perovsek.org/workflow/67/`. The abbreviations (not important for understanding the workflow) on the input and output stubs are as follows: *con* connection, *ctx* context, *odt* Orange data table, *lot* list of Orange data tables, *str* string, *arf* ARFF file, *ins* instances, *lrn* learner, *cla* classifier.

D.1 Results on the IMDB Database

The complete IMDB database is publicly available in the SQL format[1]. This database contains tables of movies, actors, movie genres, directors, director genres.

The evaluation database used in our experiments consists only of the movies whose titles and years of production exist on IMDB's top 250 and bottom 100 chart. The database therefore consisted of 166 movies, along with all of their actors, genres and directors. Movies present in the IMDB's top 250 chart were added an additional label *goodMovie*, while those in the bottom 100 were marked as *badMovie*. Additionally, attribute age was discretized; a movie was marked as *old* if it was made before 1950, *fairlyNew* if it was produced between 1950 and 2000 and *new* otherwise.

`goodMovie` ← `director_genre_drama`, `movie_genre_thriller`,
`director_name_AlfredHitchcock`.
(Support: 5.38% Confidence: 100.00%)

`movie_genre_drama` ← `goodMovie`, `actor_name_RobertDeNiro`.
(Support: 3.59% Confidence: 100.00%)

`director_name_AlfredHitchcock` ← `actor_name_AlfredHitchcock`.
(Support: 4.79% Confidence: 100.00%)

`director_name_StevenSpielberg` ← `goodMovie`, `movie_genre_adventure`,
`actor_name_TedGrossman`.
(Support: 1.79% Confidence: 100.00%)

Figure D.2: Examples of interesting association rules discovered in the IMDB database.

After preprocessing the dataset using the wordification methodology, we performed association rule learning. Frequent itemsets were generated using RapidMiner's FP-growth implementation [119]. Next, association rules for the resulting frequent itemsets were produced. Among all the discovered rules, several interesting rules were found. Figure D.2 presents some of the interesting rules selected by the expert. The first rule states that if the movie's genre is thriller and is directed by Alfred Hitchcock, who is also known for drama movies, then the movie is considered to be good. The second rule we have selected

[1]`http://www.webstepbook.com/supplements/databases/imdb.sql`

concludes that if the movie is good and Robert De Niro acts in it, then it must be a drama. The third interesting rule shows that Alfred Hitchcock acts only in the movies he also directs. The last rule concludes that if Ted Grossman acts in a good adventure movie, then the director is Steven Spielberg. Note that Ted Grossman usually plays the role of a stunt coordinator or performer.

D.2 Results on the Traffic Accident Database

The second dataset consists of all accidents that happened in Slovenia's capital Ljubljana between years 1995 and 2005. The data is publicly accessible from the national police department website[2]. The database contains the information about accidents along with all the accident's participants.

```
noInjuries ← accident_trafficDensity_rare,
            accident_location_parkingLot.
```
(Support: 0.73% Confidence: 97.66%)

```
person_gender_male ← person_vehicleType_motorcycle.
```
(Support: 0.11% Confidence: 99.12%)

Figure D.3: Examples of interesting association rules discovered in the accidents database.

The data already contained discretized attributes, so further discretization was not needed. Similarly to the IMDB database, preprocessing using wordification methodology, FP-growth itemset mining and association rule mining were performed. Figure D.3 presents some of the interesting rules found in the Slovenian traffic accidents dataset.

The first rule indicates that if the traffic is rare and the accident happened in a parking lot, then no injuries occurred. The second rule implies that whenever a motorcycle is involved in an accident, a male person is involved.

[2] http://www.policija.si/index.php/statistika/prometna-varnost

References

[1] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, USA, 2006.

[3] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[4] D. Jurafsky, *Speech & Language Processing*. Pearson Education India, 2000.

[5] C. D. Manning, P. Raghavan, H. Schütze, *et al.*, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[6] J. Cowie and W. Lehnert, "Information extraction," *Communications of the ACM*, vol. 39, no. 1, pp. 80–91, 1996.

[7] M. Alavi and D. E. Leidner, "Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues," *MIS quarterly*, pp. 107–136, 2001.

[8] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0 Step-by-step data mining guide," 2000. [Online]. Available: `http://www.crisp-dm.org/CRISPWP-0800.pdf`.

[9] M. Grčar, "Mining text-enriched heterogeneous information networks," PhD thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, 2015.

[10] S. Bird, "NLTK: The natural language toolkit," in *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, Association for Computational Linguistics (ACL), 2006, pp. 69–72.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] J. Kranjc, V. Podpečan, and N. Lavrač, "ClowdFlows: A cloud based scientific workflow platform," in *Proceedings of Machine Learning and Knowledge Discovery in Databases, ECML/PKDD (2)*, P. A. Flach, T. D. Bie, and N. Cristianini, Eds., Springer, 2012, pp. 816–819.

[13] A. Koestler, *The Act of Creation*. Hutchinson, 1964.

[14] M. Berthold, Ed., *Bisociative Knowledge Discovery*. Springer, 2012.

[15] S. Pollak, "Text classification of articles on Kenyan elections," in *Proceedings of the 4th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, 2009, pp. 229–233.

[16] S. Pollak, R. Coesemans, W. Daelemans, and N. Lavrač, "Detecting contrast patterns in newspaper articles by combining discourse analysis and text mining," *Pragmatics*, vol. 21, no. 4, pp. 674–683, 2013.

[17] M. Juršič, B. Cestnik, T. Urbančič, and N. Lavrač, “Cross-domain literature mining: Finding bridging concepts with CrossBee,” in *Proceedings of the 3rd International Conference on Computational Creativity*, 2012, pp. 33–40.

[18] M. Juršič, B. Cestnik, T. Urbančič, and N. Lavrač, “HCI empowered literature mining for cross-domain knowledge discovery,” in *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, Springer, 2013, pp. 124–135.

[19] M. Juršič, “Text mining for cross-domain knowledge discovery,” PhD thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, 2015.

[20] S. Muggleton, Ed., *Inductive Logic Programming.* Academic Press, London, 1992.

[21] N. Lavrač and S. Džeroski, “Inductive logic programming,” Ellis Horwood, 1994.

[22] S. Džeroski and N. Lavrač, Eds., *Relational Data Mining.* Springer, 2001.

[23] L. De Raedt, *Logical and Relational Learning*, L. D. Raedt, Ed. Springer, 2008.

[24] N. Lavrač, S. Džeroski, and M. Grobelnik, “Learning nonrecursive definitions of relations with LINUS,” in *Proceedings of the European Working Session on Learning (EWSL-91); Porto, Portugal*, 1991, pp. 265–281.

[25] S. Kramer, B. Pfahringer, and C. Helma, “Stochastic propositionalization of non-determinate background knowledge,” in *Proceedings of the 8th International Conference on Inductive Logic Programming, ILP-98 Madison, Wisconsin, USA, 1998*, 1998, pp. 80–94.

[26] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[27] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, “Yale: Rapid prototyping for complex data mining tasks,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, Eds., Philadelphia, PA, USA, 2006, pp. 935–940.

[28] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, *KNIME: The Konstanz Information Miner.* Springer, 2008.

[29] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3. Amsterdam: Morgan Kaufmann, 2011.

[30] J. Demšar, B. Zupan, G. Leban, and T. Curk, “Orange: From experimental machine learning to interactive data mining,” in *Proceedings of Knowledge Discovery in Databases, PKDD*, 2004, pp. 537–539.

[31] M. Abouelhoda, S. A. Issa, and M. Ghanem, “Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support,” *BMC bioinformatics*, vol. 13, no. 1, p. 1, 2012.

[32] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, *et al.*, “Galaxy: A platform for interactive large-scale genome analysis,” *Genome Research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[33] K. Okonechnikov, O. Golosova, M. Fursov, *et al.*, “Unipro UGENE: A unified bioinformatics toolkit,” *Bioinformatics*, vol. 28, no. 8, pp. 1166–1167, 2012.

[34] M. Senger, P. Rice, and T. Oinn, “Soaplab: A unified Sesame door to analysis tools,” in *UK e-Science All Hands Meeting*, National e-Science Centre, 2003, pp. 509–513.

[35] D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn, “Taverna: A tool for building and running workflows of services,” *Nucleic Acids Research*, vol. 34, no. Web-Server-Issue, pp. 729–732, 2006.

[36] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, R. Lopez, and C. A. Goble, “Biocatalogue: A universal catalogue of web services for the life sciences,” *Nucleic Acids Research*, vol. 38, no. Web-Server-Issue, pp. 689–694, 2010.

[37] D. D. Roure, C. Goble, and R. Stevens, “The design and realisation of the myExperiment virtual research environment for social sharing of workflows,” *Future Generation Computer Systems*, vol. 25, pp. 561–567, 2009.

[38] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” PhD thesis, University of California, Irvine, 2000.

[39] Y. Kano, P. Dobson, M. Nakanishi, J. Tsujii, and S. Ananiadou, “Text mining meets workflow: Linking U-Compare with Taverna,” *Bioinformatics*, vol. 26, no. 19, pp. 2486–2487, 2010.

[40] M. Poch, A. Toral, O. Hamon, V. Quochi, and N. Bel, “Towards a user-friendly platform for building language resources based on web services,” in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*, Istanbul, Turkey: European Language Resources Association (ELRA), 2012, pp. 1156–1163.

[41] M. Poch and N. Bel, “Interoperability and technology for a language resources factory,” in *Proceedings of the Workshop on Language Resources, Technology and Services in the Sharing Paradigm*, Chiang Mai, Thailand: Asian Federation of Natural Language Processing, 2011, pp. 32–40.

[42] N. Ide, P. Bonhomme, and L. Romary, “XCES: An XML-based encoding standard for linguistic corpora,” in *Proceedings of the Second International Language Resources and Evaluation Conference (LREC-2000)*, 2000, pp. 825–830.

[43] N. Ide and K. Suderman, “GrAF: A graph-based format for linguistic annotations,” in *Proceedings of the Linguistic Annotation Workshop*, ser. LAW ’07, Prague, Czech Republic: Association for Computational Linguistics (ACL), 2007, pp. 1–8.

[44] V. Arranz and O. Hamon, “On the way to a legal sharing of web applications in NLP,” in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*, Istanbul, Turkey: European Language Resources Association (ELRA), 2012, pp. 2965–2970.

[45] J.-D. Kim, T. Ohta, Y. Tateisi, and J. ichi Tsujii, “GENIA corpus - a semantically annotated corpus for bio-textmining,” in *Bioinformatics*, vol. 19, 2003, pp. i180–182.

[46] Y. Kano, W. Baumgartner, L. McCrohon, S. Ananiadou, K. Cohen, L. Hunter, and J. Tsujii, “U-compare: Share and compare text mining tools with UIMA,” *Bioinformatics*, vol. 25, no. 15, pp. 1997–1998, 2009.

[47] R. Rak, A. Rowley, J. Carter, and S. Ananiadou, “Development and analysis of NLP pipelines in Argo,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, Association for Computational Linguistics (ACL), Sofia, Bulgaria, 2013, pp. 115–120.

[48] G. Klyne and J. J. Carroll, “Resource Description Framework (RDF): Concepts and abstract syntax,” B. Mcbride, Ed., [Online]. Available: `http://www.w3.org/TR/rdf-concepts/`.

[49] R. Rak, A. Rowley, J. Carter, R. T. B. Batista-Navarro, and S. Ananiadou, “Interoperability and customisation of annotation schemata in Argo,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, European Language Resources Association (ELRA), 2014, pp. 3837–3842.

[50] M. Hinrichs, T. Zastrow, and E. Hinrichs, “Weblicht: Web-based LRT services in a distributed eScience infrastructure,” in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta: European Language Resources Association (ELRA), 2010, pp. 489–493.

[51] U. Heid, H. Schmid, K. Eckart, and E. Hinrichs, “A corpus representation format for linguistic web services: The D-SPIN text corpus format and its relationship with ISO standards,” in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta: European Language Resources Association (ELRA), 2010, pp. 494–499.

[52] T. Ishida, *The Language Grid: Service-oriented Collective Intelligence for Language Resource Interoperability.* Springer Science & Business Media, 2011.

[53] A. Bosca, L. Dini, M. Kouylekov, and M. Trevisan, “Linguagrid: A network of linguistic and semantic services for the Italian language.,” in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey: European Language Resources Association (ELRA), 2012, pp. 3304–3307.

[54] N. Ide, J. Pustejovsky, C. Cieri, E. Nyberg, D. Wang, K. Suderman, M. Verhagen, and J. Wright, “The Language Application Grid,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland: European Language Resources Association (ELRA), 2014, pp. 22–30.

[55] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, “Building Watson: An overview of the DeepQA project,” *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[56] D. R. Swanson, “Medical literature as a potential source of new knowledge,” *Bulletin of the Medical Library Association*, vol. 78, no. 1, p. 29, 1990.

[57] N. Smalheiser and D. Swanson, “Using ARROWSMITH: A computer-assisted approach to formulating and assessing scientific hypotheses,” *Computer Methods and Programs in Biomedicine*, vol. 57, no. 3, pp. 149–154, 1998.

[58] M. Weeber, H. Klein, L. de Jong-van den Berg, R. Vos, *et al.*, “Using concepts in literature-based discovery: Simulating swanson's raynaud–fish oil and migraine–magnesium discoveries,” *Journal of the American Society for Information Science and Technology*, vol. 52, no. 7, pp. 548–557, 2001.

[59] D. Hristovski, B. Peterlin, J. A. Mitchell, and S. M. Humphrey, “Using literature-based discovery to identify disease candidate genes,” *International Journal of Medical Informatics*, vol. 74, no. 2, pp. 289–298, 2005.

[60] M. Yetisgen-Yildiz and W. Pratt, “Using statistical and knowledge-based approaches for literature-based discovery,” *Journal of Biomedical Informatics*, vol. 39, no. 6, pp. 600–611, 2006.

[61] A. Holzinger, P. Yildirim, M. Geier, and K.-M. Simonic, “Quality-based knowledge discovery from medical text on the web,” in *Quality Issues in the Management of Web Information*, Springer, 2013, pp. 11–13.

[62] A. Kastrin, T. C. Rindflesch, and D. Hristovski, “Link prediction on the semantic MEDLINE network,” in *Proceedings of the 17th International Conference Discovery Science (DS 2014), Bled, Slovenia*, Springer, 2014, pp. 135–143.

[63] D. R. Swanson, “Migraine and magnesium: Eleven neglected connections,” *Perspectives in Biology and Medicine*, vol. 78, no. 1, pp. 526–557, 1988.

[64] R. K. Lindsay and M. D. Gordon, “Literature-based discovery by lexical statistics,” *Journal of the American Society for Information Science and Technology*, no. 1, pp. 574–587, 1999.

[65] P. Srinivasan, “Text mining: Generating hypotheses from MEDLINE,” *Journal of the American Society for Information Science and Technology*, vol. 55, no. 5, pp. 396–413, 2004.

[66] I. Petrič, B. Cestnik, N. Lavrač, and T. Urbančič, “Outlier detection in cross-context link discovery for creative literature mining,” *The Computer Journal*, vol. 55, no. 1, pp. 47–61, 2012.

[67] B. Sluban, M. Juršič, B. Cestnik, and N. Lavrač, “Exploring the power of outliers for cross-domain literature mining,” in *Bisociative Knowledge Discovery*, M. Berthold, Ed., Springer, 2012, pp. 325–337.

[68] T. Urbančič, I. Petrič, B. Cestnik, and M. Macedoni-Lukšič, “Literature mining: Towards better understanding of autism,” in *Proceedings of the 11th Conference on Artificial Intelligence in Medicine*, Springer, 2007, pp. 217–226.

[69] I. Petrič, T. Urbančič, B. Cestnik, and M. Macedoni-Lukšič, “Literature mining method RaJoLink for uncovering relations between biomedical concepts,” *Journal of Biomedical Informatics*, vol. 42, no. 2, pp. 219–227, 2009.

[70] M. Juršič, B. Cestnik, T. Urbančič, and N. Lavrač, “Bisociative literature mining by ensemble heuristics,” in *Bisociative Knowledge Discovery*, M. Berthold, Ed., Springer, 2012, pp. 338–358.

[71] A. Srinivasan, *Aleph manual*, 2007. [Online]. Available: `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/`.

[72] S. Muggleton, “Inverse entailment and Progol,” *New Generation Computing, Special issue on Inductive Logic Programming*, vol. 13, no. 3-4, pp. 245–286, 1995.

[73] P. A. Flach and N. Lachiche, “1BC: A first-order Bayesian classifier,” in *Proceedings of the 9th International Workshop on Inductive Logic Programming*, 1999, pp. 92–103.

[74] D. Michie, S. Muggleton, D. Page, and A. Srinivasan, “To the international computing community: A new East-West challenge,” Technical report, Oxford University Computing Laboratory, Oxford, UK, Tech. Rep., 1994.

[75] F. Železný and N. Lavrač, “Propositionalization-based relational subgroup discovery with RSD,” *Machine Learning*, vol. 62, no. 1-2, pp. 33–63, 2006.

[76] O. Kuželka and F. Železný, “Block-wise construction of tree-like relational features with monotone reducibility and redundancy,” *Machine Learning*, vol. 83, no. 2, pp. 163–192, 2011.

[77] N. Lavrač and P. A. Flach, "An extended transformation approach to Inductive Logic Programming," *ACM Transactions on Computational Logic*, vol. 2, no. 4, pp. 458–494, 2001.

[78] M. A. Krogel and S. Wrobel, "Transformation-based learning using multirelational aggregation," in *Proceedings of the 11th International Conference (ILP 2001), Strasbourg, France*, C. Rouveirol and M. Sebag, Eds., 2001, pp. 142–155.

[79] M. A. Krogel, S. Rawles, F. Železný, P. A. Flach, N. Lavrač, and S. Wrobel, "Comparative evaluation of approaches to propositionalization," in *Proceedings of the 13th International Conference on Inductive Logic Programming (ILP 2003), Szeged, Hungary*, T. Horváth, Ed., ser. Lecture Notes in Computer Science, vol. 2835, Springer, 2003, pp. 197–214.

[80] A. J. Knobbe, Ed., *Multi-Relational Data Mining*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2005, vol. 145.

[81] M. Ceci and A. Appice, "Spatial associative classification: Propositional vs structural approach," *Journal of Intelligent Information Systems*, vol. 27, no. 3, pp. 191–213, 2006.

[82] M. Ceci, A. Appice, and D. Malerba, "Emerging pattern based classification in relational data mining," in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, S. S. Bhowmick, J. Küng, and R. Wagner, Eds., vol. 5181, Springer Berlin Heidelberg, 2008, pp. 283–296.

[83] H. Guo and H. Viktor, "Multirelational classification: A multiple view approach," *Knowledge and Information Systems*, vol. 17, no. 3, pp. 287–312, 2008.

[84] M. França, G. Zaverucha, and A. d'Avila Garcez, "Fast relational learning using bottom clause propositionalization with artificial neural networks," *Machine Learning*, vol. 94, no. 1, pp. 81–104, 2014.

[85] A. Avila Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence*, vol. 11, no. 1, pp. 59–77, 1999.

[86] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of Database Systems*, Springer, 2009, pp. 532–538.

[87] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ACM, 2010, pp. 441–450.

[88] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[89] S. Scott and S. Matwin, "Feature engineering for text classification," in *Proceedings of 16th International Conference on Machine Learning (ICML'99)*, vol. 99, 1999, pp. 379–388.

[90] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the Workshop on Speech and Natural Language*, Association for Computational Linguistics (ACL), 1992, pp. 112–116.

[91] M. Juršič, I. Mozetič, T. Erjavec, and N. Lavrač, "Lemmagen: Multilingual lemmatisation with induced ripple-down rules," *Journal of Universal Computer Science*, vol. 16, no. 9, pp. 1190–1214, 2010.

[92] D. Hiemstra, "A probabilistic justification for using TF-IDF term weighting in information retrieval," *International Journal on Digital Libraries*, vol. 3, no. 2, pp. 131–139, 2000.

[93] G. Paltoglou and M. Thelwall, “A study of information retrieval weighting schemes for sentiment analysis,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics (ACL), 2010, pp. 1386–1395.

[94] A. Y. Ng, “Feature selection, L1 vs. L2 regularization, and rotational invariance,” in *Proceedings of the Twenty-first International Conference on Machine Learning*, ACM, 2004, pp. 78–86.

[95] B. Sluban, D. Gamberger, and N. Lavrač, “Ensemble-based noise detection: Noise ranking and visual performance evaluation,” *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 265–303, 2014.

[96] B. Sluban and N. Lavrač, “ViperCharts: Visual performance evaluation platform,” in *Proceedings of Machine Learning and Knowledge Discovery in Databases*, Springer, 2013, pp. 650–653.

[97] M. Perovšek, A. Vavpetič, and N. Lavrač, “A wordification approach to relational data mining: Early results,” in *Late Breaking Papers of the 22nd International Conference on Inductive Logic Programming*, 2012, pp. 56–61. [Online]. Available: `http://ceur-ws.org/Vol-975/paper-06.pdf`.

[98] M. Perovšek, A. Vavpetič, B. Cestnik, and N. Lavrač, “A wordification approach to relational data mining,” in *Proceedings of the 16th International Conference, Discovery Science 2013*, Springer, 2013, pp. 141–154.

[99] A. Vavpetič and N. Lavrač, “Semantic subgroup discovery systems and workflows in the SDM-toolkit,” *The Computer Journal*, vol. 56, no. 3, pp. 304–320, 2013.

[100] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski, “Subgroup discovery with CN2-SD,” *Journal of Machine Learning Research*, vol. 5, pp. 153–188, 2004.

[101] N. Ide and K. Suderman, “GrAF: A graph-based format for linguistic annotations,” in *Proceedings of the Linguistic Annotation Workshop, Prague, June 2007*, Association for Computational Linguistics (ACL), 2007, pp. 1–8.

[102] C. E. Brodley and M. A. Friedl, “Identifying mislabeled training data,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[103] B. Sluban, S. Pollak, R. Coesemans, and N. Lavrač, “Irregularity detection in categorized document corpora.,” in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*, 2012, pp. 1598–1603.

[104] G. Garriga, E. Junttila, and H. Mannila, “Banded structure in binary matrices,” *Knowledge and Information Systems*, vol. 28, no. 1, pp. 197–226, 2011.

[105] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo, *et al.*, “Fast discovery of association rules.,” *Advances in Knowledge Discovery and Data Mining*, vol. 12, no. 1, pp. 307–328, 1996.

[106] W. Dubitzky, T. Kötter, O. Schmidt, and M. R. Berthold, “Towards creative information exploration based on Koestler’s concept of bisociation,” in *Bisociative Knowledge Discovery*, M. Berthold, Ed., Springer, 2012, pp. 11–32.

[107] F. Alqadah, R. Bhatnagar, and A. Jegga, “Mining maximally banded matrices in binary data,” in *Proceedings of the 10th SIAM International Conference on Data Mining (SDM’10)*, 2010, pp. 942–953.

[108] J. Atkins, E. Boman, and B. Hendrickson, “A spectral algorithm for seriation and the consecutive ones problem,” *SIAM Journal on Computing*, vol. 28, no. 1, pp. 297–310, 1998.

[109] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[110] F. J. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms.," in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998)*, 1998, pp. 445–453.

[111] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[112] A. Srinivasan, S. Muggleton, R. King, and M. Sternberg, "Mutagenesis: ILP experiments in a non-determinate biological domain," in *Proceedings of the 4th International Workshop on Inductive Logic Programming*, S. Wrobel, Ed., ser. GMD-Studien, vol. 237, G, 1994, pp. 217–232.

[113] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," 1993, pp. 1022–1029.

[114] A. Srinivasan, R. D. King, S. Muggleton, and M. J. Sternberg, "Carcinogenesis predictions using ILP," in *Proceedings of the 7th International Workshop on Inductive Logic Programming (ILP-97), Prague, Czech Republic*, ser. Lecture Notes in Computer Science, vol. 1297, Springer, 1997, pp. 273–287.

[115] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[116] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 675–701, 1937.

[117] P. B. Nemenyi, "Distribution-free multiple comparisons," PhD thesis, Princeton University, 1963.

[118] J. Demšar, "Statistical comparison of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[119] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'06), New York, NY, USA*, ACM Press, 2006, pp. 935–940.

Bibliography

Publications Related to the Thesis

Journal Articles

M. Perovšek, J. Kranjc, T. Erjavec, B. Cestnik, and N. Lavrač, "TextFlows: A visual programming platform for text mining and natural language processing," *Science of Computer Programming: Special Issue on Knowledge-based Software Engineering*, vol. 121, pp. 128–152, 2016.

M. Perovšek, A. Vavpetič, J. Kranjc, B. Cestnik, and N. Lavrač, "Wordification: Propositionalization by unfolding relational data into bags of words," *Expert Systems with Applications*, vol. 42, no. 17, pp. 6442–6456, 2015.

Conference Papers

M. Perovšek, B. Cestnik, T. Urbančič, S. Colton, and N. Lavrač, "Towards narrative ideation via cross-context link discovery using banded matrices," in *Proceedings of the 12th International Conference on Intelligent Data Analysis*, Springer, 2013, pp. 333–344.

M. T. Llano, R. Hepworth, S. Colton, J. Gow, J. Charnley, N. Lavrač, M. Žnidaršič, M. Perovšek, M. Granroth-Wilding, and S. Clark, "Baseline methods for automated fictional ideation," in *Proceedings of the International Conference on Computational Creativity*, 2014, pp. 211–219.

M. Perovšek, N. Lavrač, and B. Cestnik, "Bridging term discovery for cross-domain literature mining," in *Proceedings of the 17th International Conference on Discovery Science (DS 2014): Book of Abstracts*, 2014. [Online]. Available: `http://ds2014.ijs.si/lbp/DS2014_LBP_complete.pdf`.

M. Perovšek, V. Podpečan, J. Kranjc, T. Erjavec, S. Pollak, N. Q. Do Thi, X. Liu, C. Smith, M. Cavazza, and N. Lavrač, "Text mining platform for NLP workflow design, replication and reuse," in *Proceedings of IJCAI Workshop on Replicability and Reusability in Natural Language Processing: From Data to Software Sharing*, 2015. [Online]. Available: `https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhZGFwdGl2ZW5scDIwMTV8Z3g6NTU1M2Q5MmYzZTYyNTZkNQ`.

M. Perovšek, A. Vavpetič, and N. Lavrač, "A wordification approach to relational data mining: Early results," in *Late Breaking Papers of the 22nd International Conference on Inductive Logic Programming*, 2012, pp. 56–61. [Online]. Available: `http://ceur-ws.org/Vol-975/paper-06.pdf`.

M. Perovšek, A. Vavpetič, B. Cestnik, and N. Lavrač, "A wordification approach to relational data mining," in *Proceedings of the 16th International Conference, Discovery Science 2013*, Springer, 2013, pp. 141–154.

N. Lavrač, M. Perovšek, and A. Vavpetič, “Propositionalization online,” in *Proceedings of the European Conference, ECML PKDD 2014, Nancy, France*, Springer, 2014, pp. 456–459.

Other Publications

Conference Papers

M. Perovšek, N. Lavrač, and B. Cestnik, “Visual divisive hierarchical clustering using k-means,” in *Proceedings of the 15th International Multiconference Information Society - IS 2012, October 8th-12th, Ljubljan, Slovenia*, 2012, pp. 193–196.

Biography

Matic Perovšek was born on December 26, 1987 in Ljubljana, Slovenia. He finished his primary and secondary education in Grosuplje and Ljubljana, Slovenia. In 2006 he started his studies at the Faculty of Computer Science, University of Ljubljana. In 2011 he finished the undergraduate programme by defending his BSc thesis entitled "Single-class recommendation systems" under the supervision of Prof. Dr. Blaž Zupan.

In 2012 he was accepted to the position of a young researcher at the Jožef Stefan Institute, Slovenia, under the supervision of Prof. Dr. Nada Lavrač. In the same year he started his graduate studies at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia under the supervision of Prof. Dr. Bojan Cestnik and co-supervision of Prof. Dr. Nada Lavrač. He is enrolled in the PhD programme "Information and Communication Technologies".

His research in the field of computer science mostly focuses on developing novel cloud-based approaches to text mining and natural language processing using graphical scientific workflows and developing applications and infrastructures to facilitate literature-based cross-context discovery. The applications of the developed platform and workflows span several fields such as text mining, natural language processing, and relational data mining. He presented his work at international conferences and workshops, and published several journal papers.