Faculty of Technology,
Semantic Computing Group

Master's Thesis

Ranking of Disease Gene Associations from large Corpora of Scientific Publications

Hendrik ter Horst

Supervisors:

Matthias Hartung, M.A.
Prof. Dr. Philipp Cimiano

Date: 26th June 2015

Universität Bielefeld
Universitätsstraße 21–23
33615 Bielefeld

Abstract

The extraction of disease-gene associations from biomedical publications is a widely investigated field of research. In previous work, a frequent method was to implement natural language processing tools that use semantic information to find such associations. However, most of these approaches are restricted to single documents. Retrieval systems that predict novel associations across various documents often lack the ability to deal with the huge amount of resulting candidates. In this work, we present a system that aggregates information from a large corpora of scientific abstracts. This information is used to build a comprehensive gene-interaction network, which is then used to predict novel disease-gene associations. We tackle the problem of candidate reduction by integrating two separate machine learning methods. We train a support vector machine to classify genes as disease related or not and a support vector regression model to rank gene-candidates according to their importance to a specific disease. Thereto, we make use of approved methods and extend them by a novel investigation of the gene-interaction network. In a model-evaluation on two gold standards as well as in a case-study in cooperation with biomedical experts, it is shown that the proposed methods are able to extract disease-gene-associations from single documents and discover disease-related candidates across multiple documents.

Contents

1 Introduction

1.1 Motivation

The huge amount of biomedical information, which is published daily exceeds the human abilities to recognize and process relevant knowledge manually. It is barely possible to extract all available information without any technical support. Due to improved technology, nowadays extraction systems can process large corpora of scientific publications for several applications. One very well investigated research field is the automated discovery of disease-gene associations.

Finding causal links between biomedical entities e.g. between diseases and genes, receives a growing attention in the recent years. Fundamental sub-tasks like named entity recognition, entity disambiguation and relation extraction have been tackled very successfully. Improvements have been made in collecting and providing already extracted information by storing them standardized in global databases. The exponential growth of research results provides an essential groundwork for nowadays approaches and with the increasing importance of machine learning methods, the amount of annotated corpora were enhanced in quantity and quality. However, the frequent emergence of new challenges, for example the BioCreative shared tasks [33, 35, 47], shows the deep and lasting interest of the biomedical research community.

Due to the growing popularity of Swanson's fish-oil experiment [64], the task of relation extraction enlarged. Swanson was able to link researched information across two publications and discovered new knowledge. In his work, he defined existing information that can be found in a single publication as *private knowledge* and information that can only be gathered by linking private information as *(undiscovered) public knowledge.* This encouraged many researchers to develop novel approaches that deal with the discovery of public knowledge. However, the results of systems that automatically extract public knowledge, often need further investigations and a subsequent human assessment. Especially the task of disease associated gene extraction, using public knowledge, should rather be seen as the generation of disease related gene-candidates. A common way of public knowledge discovery is to built interaction networks of the studied concepts. These networks tend to grow very fast, which also increases their complexity. This results in a new problem. The number of predicted candidates rises exponentially to the size and complexity of the network. This makes a human based assessment a lot more expensive and time consuming.

1.2 Goals And Contributions

In this work, we present a comprehensive system for disease-gene association extraction as well as disease-related gene-candidate prediction with respect to the human species.

Our approach consists of a modular pipeline, built of several state-of-the-art systems for entity recognition, disambiguation, and gene-event extraction from a large corpora of scientific publications. Besides that, we implemented a simple disease normalization framework (*DiNo*). We provide a simplified GENIA-event-scheme in order to generalize event-types. Based on the extracted information, we built a large gene-interaction network by integrating the transformed gene-events into a graph-database. We use this network to generate gene-candidates for a specified disease. Since the network was built by an aggregation of information across various publications, the candidate generation goes beyond private knowledge and allows us to discover novel disease-gene associations (hidden public knowledge).

We tackle the problem of candidate reduction by implementing two machine learning methods. A support vector machine is trained to classify gene-candidates as disease-related or not, and a support vector regression model is trained to rank genes by their importance. Both methods can be used to filter gene-candidates by discarding genes that are either non-related or whose prediction is lower than a threshold. Our feature space consists of already proved features based on disease-gene co-occurrences and is extended by novel features based on the interaction network.

We evaluate our models on two different gold-standards. Further, we evaluate our system by integrating the best performing models and conduct a case-study. We chose *Pulmonary Fibrosis* as testing diseases.

Our contribution to the research field is a comprehensive system, that generates a list of disease related gene-candidates across scientific publications ranked by their importance.

1.3 Outline

After the brief introduction and motivation of our presented work, we give an outline of the remaining chapters of this thesis.

In **Chapter 2**, we present related work to our approach. We briefly introduce some state-of-the-art systems that deal with sub-tasks such as *Named Entity Recognition*, *Entity Disambiguation*, and *Gene-Event Extraction.* Besides that, we present existing approaches that tackle the comprehensive task of disease-gene association extraction and systems for candidate generation.

Chapter 3 gives a brief overview of the existing resources that we used in this work. This includes two gold-standards, the large corpora of scientific publications, and several databases for the task of entity disambiguation.

Chapter 4 contains a description of the overall system architecture. We provide a comprehensive view to the modular structure. Each module is then described more detailed with respect to its input and output. We illustrate the functional interaction between the modules and explain the chain of data processing.

Chapter 5 presents all methods that we use in our approach. This includes integrated state-of-the-art systems, their functional combination, as well as their combination with self developed methods. We explain the disease recognition and normalization, the gene-event extraction, which subsumed the gene recognition, the gene normalization and event transformation. On top of that, we motivate and describe the data storage and data supply, which is part of the resource generation and enables the gene-interaction network. In the last section, we formalize the features, which are used in the machine learning methods.

In **Chapter 6**, we describe the conducted experiments and present the evaluation results. We test various settings against two different gold-standards to determine the best performing models. These models are used in a case-study to test the performance of the gene-classification task. We present and discuss these evaluation results as well.

Chapter 7 summarizes the results of this thesis with respect to the aim of our contribution. We briefly discuss the evaluation results and give an overview of remaining problems. The discussion contains some ideas for future work that can help to improve the system.

2 Related Work

We situated our work on *Ranking of Disease Gene Associations from large Corpora of Scientific Publications* in the context of entity relation extraction.

Entity relation extraction (ERE) is the task of identifying relations or complex events between entities, such as diseases, genes or proteins, in natural language texts but also in existing databases. This includes finding relations between entities of the same type, e.g. relations between genes, but also across various types. Two frequent types of ERE, especially in the biomedical research field, are the extraction of gene relations among each other (GR) and the extraction of disease-associated genes (DAG). Both of them share common problems like Named Entity Recognition (NER) and disambiguation of such named-entities (NED). These problems have been addressed relatively successful in recent years.

Named Entity Relation Facing the challenges of NER, e.g. the unseen word problem or the mention boundary problem, BANNER [39] turns out to be one of the best performing approaches. It uses state-of-the-art machine learning techniques to maximize domain independence. Like other NER approaches [36, 66], BANNER defines the named entity recognition as a label sequence problem and tackles it by using conditional random fields (CRF). This combination makes BANNER a powerful framework for extracting disease, genes and other biomedical entities. In 2008 Dingcheng Li et al. compared CRF with support vector machines (SVM) for the task of named entity recognition in the biomedical field. They showed that CRFs outperform SVMs by 22 points in F-score [43] with a total of 86%.

Named Entity Disambiguation Most of these state-of-the-art approaches extract the surface form[1] of an entity and its position, but do not take the process of NED into account. However, this is a crucial step, especially if the extracted information need to be further processed, e.g. in ERE-systems and should be linked to structured information

[1]The surface form is defined as the unmodified form of how the entity appears in a text.

in databases or otherwise harmonized with existing knowledge from different sources. Such normalizations are not restricted to approaches that make use of natural language processes. In contrast, due to the establishment of many different so called standards for biomedical named entities, its a common way for both, automatically generated databases and curated databases to use self defined standards. This leads to a high variation of surface forms, preferred names and IDs [12, 45, 67]. Thus, a disambiguation is mandatory [2, 11, 68].

Robert Leaman et al. have introduced DNorm [38], which tackles the problem of disease name normalization. DNorm makes use of machine learning methods, that learn similarities between mentions and concept names. A common technique in disease normalization is to make use of databases such as MeSH [44] and OMIM [31] with predefined terms and concept names for the specific domain [23].

This structure applies also for different types of concepts such as genes. In 2009, Joachim Wermter et al. challenged the task of gene name normalization and presented GeNo [72]. They illustrated that the task of normalization and disambiguation does not only rely on machine learning but also on dictionary string matching techniques. The mixture of both techniques led to an F-score of 86.4 points. However, GeNo uses a build-in named entity recognition module which is nowadays a common method [38, 71, 72]. It offers the possibility to influence the NER-output and customize the normalization to this specific output. Nonetheless, Haw-ren Fang et al. showed that taking NER and entity normalization separated into account works, too [25].

Jörg Hakenberg et al. challenged a more comprehensive task by not specializing to a specific taxonomic scope [28]. The task of gene normalization across various species requires more than a dictionary based string matching and machine learning techniques. Especially working with inter-species gene names is a complicated challenge. Hakenberg presented GNAT, a publicly available system for inter-species gene name normalization. It shows an performance of 81.4 F-score, taking 13 species into account. However, all the adjustments to deal with the problems of inter species genes, lead to a slightly drop in the performance for individual species. Overall, they provided an F-score of 85.4% for human gene normalization.

Relation Extraction Named entity recognition and entity normalization are both elementary components of relation extraction systems e.g. for gene relations or gene event extraction and disease-gene association extraction. Finding gene relations in biomedical literatures is a well established research field. There are various approaches that implement many different methods. Claudio Giuliano et al. made use of shallow linguistic information to extract relations between known entities. They implemented two kernel functions [58] to identify useful information about local and global structures of a sentence [27].

Katrin Fundel et al. used natural language processing to produce syntactic parse trees. Using this additional layer of linguistic information, they showed, that they were able to extract gene-protein relations from a large corpora of scientific abstracts. Fundel mentioned the importance of an integrated normalization in order to provide additional information from various databases [26].

Event Extraction Gene relation extraction is an important challenge for automatic discovery of interactive biomedical entities. Jari Björne et al noticed a shift from relation extraction to the more complex task of event extraction [8]. In contrast to the relation, an event provides more than just a binary or fuzzy interaction [73] type. An event contains detailed information about the specific interaction like the sentence structure and the direction of the relation. This could be either positive, neutral or negative. On top of that, an event could describe a hierarchical structure of multiple relations.

One of the most popular event extraction systems for gene interactions is TEES, [9]. TEES is a highly competitive system that was successfully applied to various BioCreative shared tasks on event extraction [62]. The system's pipeline is sub-modular designed. A pre-processing step contains various state-of-the-art systems for sentence parsing and named entity recognition. This modular structure makes TEES a powerful framework. It benefits not only from self made improvements for the actual event recognition task but also by each local improvement in individual modules.

Gene Interaction Network Sofie Van Landeghem and Jari Björne applied TEES to the entire PubMed and built a protein/gene interaction network [69]. The main goal was to create a comprehensive network for further research. To reduce the amount of data and for greater usability they normalized both the extracted proteins/genes and the extracted events. Landeghem created a network with more than 70 million protein/gene mentions and 40 million events connecting them for 5.032 species across the full taxonomic scope.

Disease Gene Associations The extraction of disease associated genes (DAG) is a complex task that got increased attention during the recent years. Especially for human disorders, a lot of research was done and a huge amount of knowledge has been aggregated. Swanson et al. differentiate between private and public knowledge [65]. They defined private knowledge as information that can be found within a single publication. In contrast, public knowledge is gathered by combining the results from various publications. In their popular fish-oil experiment [64] they proved this hypothesis. They showed that two complementary publications could reveal useful information. This knowledge may not exist in a single publication alone. Their findings motivated several researchers to face the problem to find undiscovered public knowledge in both biomedical research fields and beyond that.

Dimitar Hristovski et al. introduced BITOLA, an interactive support system for biomedical discovery of new, hypothetical relations between biomedical concepts [6]. In fact they use their system to discover disease-gene associations by mining the literature. The main idea is: if concept A is related to concept B and B is related to concept C than A is potentially related to C, too. This is a common way to predict new potential candidates for entity relation [32].

Another method to extract novel DAG candidates was introduced by Jonathan D. Wren et al. They used text mining techniques to identify biomedical concepts of interest, e.g. disease and genes and created a network of tentative relationships. These relationships between concepts are based on co-occurrence. They break down the strict binary co-occurrence relation into a fuzzy form by taking several indicators into account. The tentative network is then ranked against a random network model to estimate statistical significance of the relationships. Wren demonstrated their approach by finding an association between CPZ-gene and cardiac hypertrophy. They were able to show that building a network model has great potential for finding novel candidates.

The approach from Wilkinson et al. is build on a similar base like Wren's. They build a network of gene-gene co-occurrences, which were extracted from the literature. However, their approach differs in both the method and the goal. Wilkinson described the goal as finding gene-communities that are related to a specific disease e.g. colon cancer. These gene communities are build by using various graph algorithms to cluster the complex gene graph. The goal is to provide a list of genes which are either known as related to a the disease or serve as potentially new candidates. They were able to show that a network structure is a powerful method to organize related entities with known and maybe unknown relations.

Özgür et al., too, ascertain the benefits of an interaction network to identify disease associated genes. Their approach builds on a network that contains diseases and genes. In contrast to the previous presented approaches these interactions do not rely on co-occurrences but on text mining and dependency parsing. Özgür implemented support vector machines to learn important associations in the network based on graph analysis features like betweenness, closeness and centrality. Their goal was to rank genes within the network to infer unknown DAGs [49]. A similar approach was developed by Changqin Quan et al. They used a maximum entropy model with topic features and a probabilistic context free grammar to identify disease gene associations [53].

Besides network-based approaches that are often developed to detect both private and public knowledge, there are some approaches that restrict their information extraction to private knowledge. They can be roughly divided into text mining and interaction database based methods [53]. Hisham Al-Mubaid et al. presented a text mining approach to find associations between proteins and diseases. Next to extensive NLP and machine learning methods, a common technique is still the co-occurrence recognition in sentence or abstract level. This has several reasons. One of them is the fact that negative results are published with less frequency. Consequently, abstract based co-occurrence leads to a high recall [17] whereas sentence based co-occurrence rather leads to a high precision [52]. However, Al-Mubaid presents a new technique using information theoretic concepts like expectations, evidence and Z-score to support the co-occurrence based extraction [2].

M. A. van Driel et al. presented a web-based tool, namely *GeneSeeker* [68], for the identification of disease related genes. Their approach uses deep semantic information about biomedical entities in order to predict a relation or not. The needed information is gathered from nine different databases. Driel's web tool is specialized to human genes and thereby implements some rules for human gene identification, although they gather the information from databases. This approach requires a gene name disambiguation in order to map genes and correctly assign collected information.

Comparison In our approach, we tackle the problem of relation extraction between diseases and genes. We share the common goal to detect already known (private) relations, but lay the focus on finding undiscovered (public) relations. We define a disease-gene association as *undiscovered*, if it is not presented in a curated databases. We aim at genes that are not in CTD [22], DisGeNET [51] or GTR [54]. In the current system, we restrict our findings to human genes. However, there is no obstacle on the implementation-side, that prevents us from taking a larger or different taxonomic scope into account. Our method contains a pipeline of several state-of-the-art systems [9, 36, 72] and some new developed frameworks. This pipeline can be divided into several sub-components. At first, we extract all biomedical concepts of interest, namely diseases, genes and gene events. We use TEES [9] to extract genes, and gene events from all 22 million abstracts in PubMed. Both, gene mentions and events are then normalized. For the task of gene normalization, a modified version of GeNo [72][2] is used. In agreement with biomedical experts, we implemented an event-normalization tool that fits the GENIA event scheme [34]. The resulting normalized data is then used to create a gene interaction network similar to Landeghem in [69]. This network connects normalized genes (nodes) through normalized events (edges). Besides the gene/event extraction we integrate the CRF tagger developed by Klinger in [36] for disease name recognition. To normalize the diseaser mentions, we implemented a simple disease normalization tool, based on DNorm [38].

The actual task of gene candidate generation starts by extracting all abstract based disease-gene co-occurrences. We call these genes *base-genes*. For each base-gene that co-occurs with the specific disease, we create a set of *additional-genes* by following each possible path (outgoing from the current base-gene) in the gene network with a specific path-length. Both, base genes and additional-genes serve as gene candidates. Using interaction networks to detect new, potential genes, leads to a large number of possible candidates. We tackle the problem by integrating machine learning approaches to

1. classify genes as disease-related or not, using a support vector machine.
2. rank all candidates by their importance to the disease, using a support vector regression model.

In the first case, the result is a set of genes that are likely related to the disease. The latter case generates a list of genes sorted by their importance. We are able to apply a filter to eliminate all candidates that are already presented in one of the databases, mentioned above.

[2]The full pipeline of GeNo already implements a gene recognition tool was integrated. Since we are using TEES as recognizer we use only the gene normalization tool of GeNo. A detailed description can be found in Section 5.2.2

3 Materials

This chapter gives a brief overview of all materials that we utilized in this work. All following materials are offline versions that were downloaded at the beginning of the development in 2014. For convenience, we keep the actual name of each material, but always refer to the downloaded (and modified) data dump. Most of these materials are existing databases that are maintained by large biomedical companies like the National Center of Biotechnology Information [56] (NCBI) or the U.S. National Library of Medicine (NLM) [1] . This chapter describes six different databases that are widely used in the literature for similar tasks and accepted by the biomedical research community. These are

1. the ***Med**ical Literature Analysis and Retrieval System On**line*** [1] (MEDLINE), which is a comprehensive collection of research publications.
2. three different databases that contain biomedical concepts and provide information for the task of entity disambiguation, namely, *Online Mendelian Inheritance in Men* [31] (OMIM), *Medical Subject Headings* [44] (MeSH) and *EntrezGene* [45].
3. two curated databases of disease-gene associations. Both are used as gold standard to train our machine learning methods and evaluate our system. The *Genetic Testing Registry* [54] (GTR) contains binary disease-gene associations and is used to solve the classification task. In contrast, we use *DisGeNET* for the regression/ranking task, since DisGeNET [51] provides a fuzzy score for each disease-gene association.

3.1 The Investigated Corpora PubMed

Medical Literature Analysis and Retrieval System Online MEDLINE is a public bibliographic database maintained by the U.S. National Library of Medicine. Since the beginning of the project in 1946, the number of citations grew exponentially to a total number of 25 million abstracts (2015). MEDLINE covers a large number of biomedical fields and contains research results from several thousand scientists and physicians. Due to the enormous size, the huge variety of topics and the free accessibility, MEDLINE is one of the most important databases for biomedical researchers. The importance is reflected particularly in the growing number of related projects. This includes many established search engines [24,48,59] for purposeful queries, but also the growing number of biomedical competitions. One of the most popular set of competitions is the BioCreative shared task [3,4,46]. MEDLINE, in its size and complexity, provides an excellent basis for knowledge discovery and information aggregation.

In this work, we use a downloaded MEDLINE dump from the year 2014. A brief statistic is shown in Table 3.1. The first column contains the total number of abstracts. The

	Total Abstracts	ASCII	Non-ASCII	Considered Abstracts
Count	22.271.876	21.362.350	909.526	21.360.113

Table 3.1: Detailed statistic of the used MEDLINE data dump, downloaded in 2014.

second column contains the number of ASCII-formated abstracts. The reason to split the abstracts into ASCII and Non-ASCII is that we had issues applying TEES to the Non-ASCII abstracts, which is an already known problem[1]. We ignore all abstracts that can not be processed with our full pipeline in the current system. On top of that, MEDLINE contains different versions for some abstracts. Although it integrates a versioning index, each version is listed individually. However, each version of an abstract shares the same PubMed-ID. We assume that new versions do not change the experimental results or the semantic content of the abstract but rather correct spelling errors etc.[2]. Thus, we filter all abstracts whose PubMed-IDs occur more than once. The actual number of considered citations is shown in the column *Considered Abstracts*.

3.2 Data for Entity Disambiguation

Online Mendelian Inheritance in Men OMIM [31] , maintained by the National Center for Biotechnology Information, is a high quality database for Mendelian disorders, human genes and gene mutations. Each entry in OMIM is the result of a biomedical experiment that was curated and evaluated by human scientists. We use OMIM to create a comprehensive disease-name dictionary for our disease normalization tool described in Section 5.1.2. The downloaded data dump contains approximately 6.800 distinct diseases.

[1] `https://github.com/jbjorne/TEES/issues/11` Access: 2015-04-25

[2] `http://www.nlm.nih.gov/bsd/licensee/elements_descriptions.html` Access: 2015-04-25

Medical Subject Headings MeSH [44], maintained by the National Library of Medicine, is a large vocabulary thesaurus of biological terms with the main purpose of indexing biomedical abstracts and full-text journals. Likewise to OMIM, we use MeSH for disease normalization. MeSH consists of two different datasets, the descriptor file (Desc) and the supplemental file (Supp). Desc can be seen as the main resource. It contains thousands of medical subjects, such as diseases and symptoms, distributed in a large scope of semantic types. The supplemental file contains additional information to many records of Desc. However, in the following work we do not distinguish between Desc and Supp but always refer to the entire MeSH. The downloaded MeSH dump provides up to 16.000 diseases that we use for the disease normalization task in Section 5.1.2.

EntrezGene EntrezGene [45], maintained by NCBI, is a meta search engine that collects genes from public gene databases. The collected genes are unified and each gene is provided with a unique ID. EntrezGene provides a lot of information for each specific gene. Next to others, this includes a taxonomic identifier, a preferred gene-symbol, a list of synonyms and other nomenclatures. This very extensive storage of gene information makes EntrezGene a very famous database for gene normalization frameworks. The taxonomic ID makes it even more potent, as it can be used to filter genes that do not belong to a specific taxonomy, e.g. homo sapiens. EntrezGene distinguishes between several types of genes: *miscRNA*, *ncRNA*, *protein-coding*, *pseudo*, *rRNA*, *scRNA*, *snoRNA*, *snRNA*, *trRNA*, *other* and *unkown*. In this work, we restrict our interest to *protein-coding* genes whose taxonomic-ID is 9606 (homo sapiens). After filtering EntrezGene by applying our constrains, 20.700 human genes remain.

3.3 Gold Standards

DisGeNET The first gold data that we use is DisGeNET [51]. It is maintained by the Integrative Biomedical Informatics Group[3] (IBI) . DisGeNET is a comprehensive database that collects disease-gene associations from several public resources. These resources are public databases that contain disease-gene associations, using different methods of knowledge aggregation. Thus, DisGeNET provides a confidence score for each association. The score describes a kind of quality of the corresponding association. The more considered resources contain the specific association the higher the score. Each occurrence in a resource is weighted. The weight depends on the underlying method and the scope of the resource.

[3] `http://ibi.imim.es/` Access: 2015-04-25

This gold standard contains a total number of 381.000 disease-gene associations between 16.000 genes and 13.000 diseases. Due to providing a score, DisGeNET can be used to train regression models that are used to address the task of gene ranking.

Genetic Testing Registry The *Genetic Testing Registry* [54] (GTR) is a central database for human researchers to migrate their results from biomedical experiments. The majority of these experiments lays in the interaction between Mendelian disorders and human genes. With respect to use GTR as gold standard to train machine learning models and evaluate our system, the composition of GTR has pros and cons. On the one hand, due to the human curation, it provides a high quality data set. On the other hand, GTR has a very limited number of entries in comparison to databases whose data is automatically aggregate. Our data dump of the Genetic Testing Registry consists of 4.200 conditions, 2.800 genes and 5.800 disease-gene associations. Since each presented association is curated by human knowledge, there is no need for a confidence score qualifying the DAG. Thus we use GTR as gold-standard data for the disease-gene classification task that is described in Section 6.1.

4 System Architecture

This chapter describes the architecture of our system. It is designed as a pipeline of consecutive and parallel modules. In the following sections, we describe each module separately to convey our idea of consecutive data processing. Underlying algorithms are omitted in this chapter and will be discussed in Chapter 5. To understand the system's architecture, it suffices to lay the focus on the analysis of the input and output of each module. The goal of this chapter is to provide an overview of the entire system, the data-processing steps and to show which processes can be calculated offline and which processes need to be calculated online.

Our main motivation to design the system as a pipeline of consecutive and parallel modules is to provide a comprehensible way of data integration. This should not only effect the initial data integration but later integrations as well. This design allows us to keep the system up-to-date by simply integrating newly relevant upcoming data. The modularity opens up the possibility to update or exchange outdated approaches. One important key feature during development was a strong transparency of each module. This allows us to get an understanding of what happens inside the system and makes it easy to discover weak-points. The overall architecture is illustrated in Figure 4.1.

Information Extraction The illustration shows six separated main modules. It starts with the *Information Extraction* module located in the top left corner. In this pre-processing step, we extract all biomedical entities and information of our interest. In the current system, this includes gene-events as well as gene and disease mentions. However, the information extraction is easily expandable. In future work, we may extend the information extraction to meta-data information for each entity.

Data Conversion In the second main module, namely *Data Conversion*, we aim at the preparation of the extracted information to store them persistently in our database. This step includes the entity disambiguation/normalization, data simplification and finally the data conversion into the database readable format.

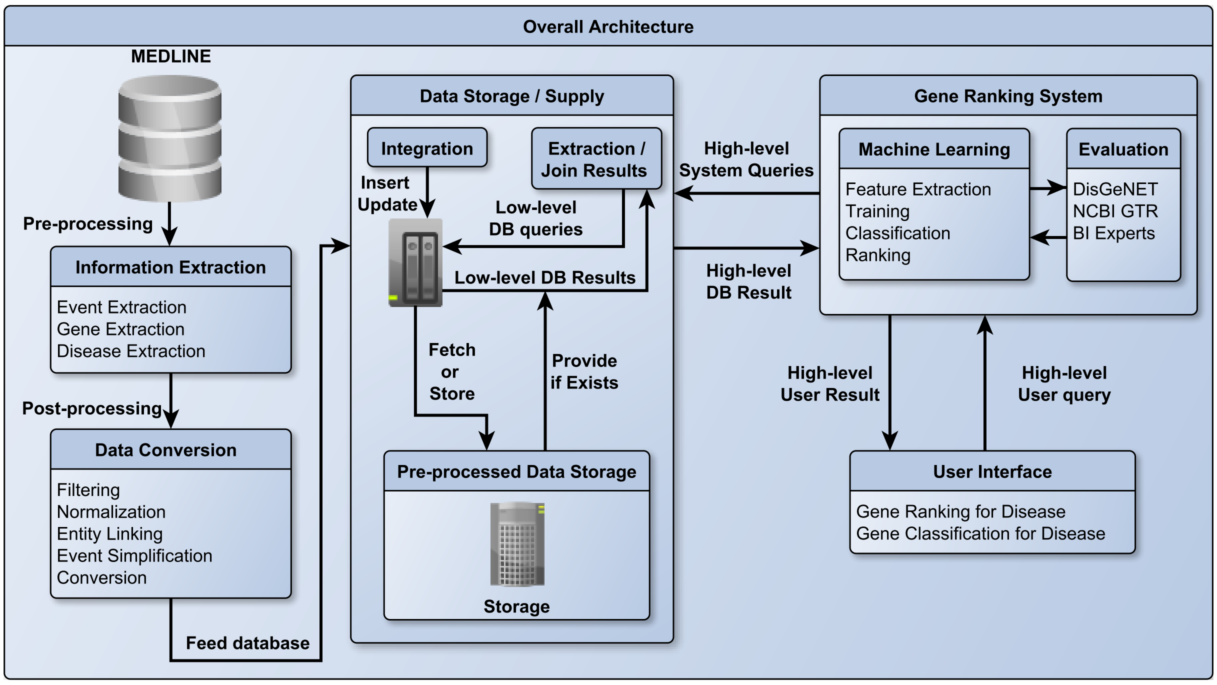

Figure 4.1: System overview.

Data Storage and Supply The task of persistent data storage by integrating the extracted information into a database and its fast supply, is shown in the *Data Storage and Supply* module. On the one hand, it handles both the initial data integration and integration of new data. On the other hand, it converts high-level queries into complex low-level queries. The extracted data is then parsed into high-level objects. It should be noted that the first three modules are mono-directional connected. Up to this point, all processes in the modules are calculated offline. However, the bidirectionally connection between the *Data Storage and Supply* module and the *Gene Ranking System* indicates an online interaction. We simulate this online interaction. In order to speed up the entire process, a lot of frequently queried information was pre-processed earlier and stored in the *Pre-processing Data Storage*. The *Pre-processing Data Storage* can be seen as an independent background module that stores already queried and processed information and provides them instantly.

Gene Ranking System Inside the *Gene Ranking System* module, we distinguish between the *Machine Learning* sub-module, that actually processes and calculates the user queries and the *Evaluation* module. The latter one is technically not part of the systems pipeline and is only used during development. However, the user interacts with the *Gene*

Ranking System module. He is able to transmit high-level user queries that are internally converted into high-level system queries. Through the bidirectional interaction with the database, it gathers all information that is need to accomplish the user query. This could be either a list of ranked genes for a specified disease or a set of genes that are classified as disease related.

In the next few sections, each main module is viewed in detail with respect to its input and output. Therefore, a more detailed illustration of each module is given, which provides a deeper insight. Most of the modules in turn consist of several sub-modules themselves. To simplify the descriptions, we use a consistent view to these illustrations. Sub-modules that are in horizontal order run consecutively. Sub-modules that are vertically stacked run in parallel.

4.1 Information Extraction

The first module in our pipeline is the *Information Extraction.* The module consists of two separated extraction frameworks, namely the *Turku Event Extraction System* [9] (TEES) and the CRF based disease tagger [36] (DT). Both frameworks contain NLP methods that can be applied to plain textual input sources. In this case the input is the MEDLINE abstract with the PubMed-ID 23848600. We apply TEES and DT in parallel to the input source. The output of the disease extraction is a file that contains a list of disease mentions, found in the input text. Each mention is provided with the disease surface form, the position within the abstract, a sentence identifier in which the disease was found and the PubMed-ID. The gene/event extraction sub-module generates a file that contains each gene mention and each extracted event. Similar to the previous file, each found entity is provided with the surface form, the position within the abstract and the PubMed-ID. However, TEES does not support a sentence identification. The entire module with all its sub-modules for the task of information extraction is visualized in Figure 4.2

4.2 Data Conversion

The second module is the *Data Conversion.* The objective of this module is to process the previously extracted information in order to transform it into a database friendly format. The input of this module are the two previously generated files. First, the disease

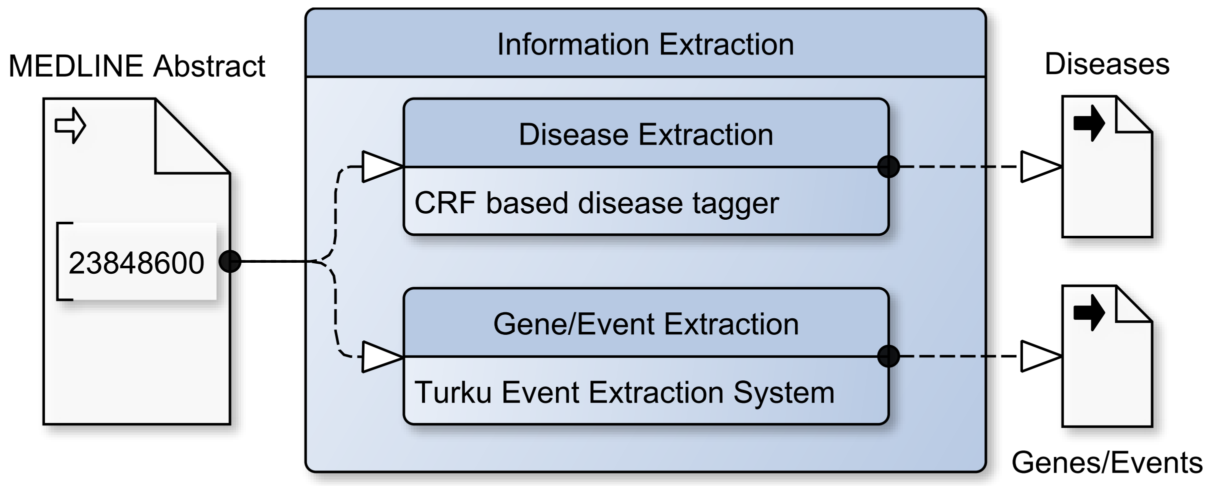

Figure 4.2: Detailed visualization of the *Information Extraction* module from Figure 4.1 The input is an MEDLINE abstract with the PubMed-ID 23848600. The output are two independent files that contain the extracted information.

mentions and the gene mentions are normalized. Both normalizations are independent sub-tasks. Thus, both processes can run in parallel. We use *DiNo* (described in Section 5.1.2) for the disease normalization and *GeNo* (described in Section 5.2.2) for the gene normalization. All normalized genes are then passed to the *Filtering* module where all non human-genes are filtered. Both, the normalized diseases and the normalized human genes are passed to the *Entity Linking* module. It creates the first information type that needs to be stored in the database, namely the abstract-based *disease-gene co-occurrences* (DGC). In parallel, the *Event Simplification* module converts complex gene events into simple gene events. It should be noted that only those gene events are persisted, whose affected genes are human genes.

Both, simple gene events and DGCs are passed to the *RDF Generation.* This sub-module converts its input into the database readable format, namely RDF-triple. The output is a comprehensive list of triple that were created from both input sources. The entire module with all its sub-modules is illustrated in Figure 4.3

4.3 Data Storage and Supply

The third module is the *Data Storage and Supply.* In contrast to the previous two models, this module contains a bidirectional connection to its next module. The main objective of *Data Storage and Supply* is to store previously extracted and converted information

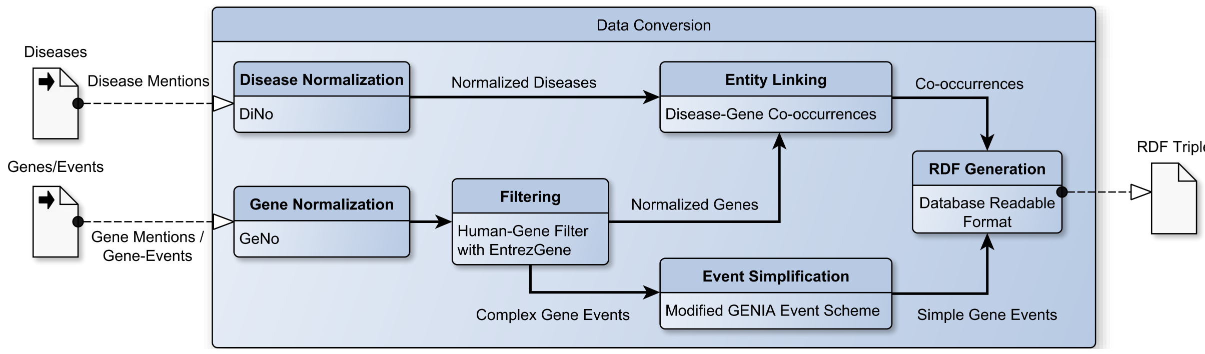

Figure 4.3: Detailed visualization of the *Data Conversion* module. The inputs are the previously generated files that contain the extracted information. The output is a comprehensive list of RDF-triples covering the extracted information.

persistently and provide them at query time. In the first step, all created RDF triples are used as input. The sub-module *Integration* determines if the triples need to be updated or not. If the database is empty, it prepares the data for a batch insertion, otherwise the preparation happens with respect to an updating function. The second input source comes from the *Gene Ranking System* in form of a high-level system query (HLSQ). If the requested HLSQ was already queried, the *Extraction* module passes the query to the *Storage*. The *Storage* outputs the previously processed high-level database result (HLDR). If not, the HLSQ is converted into several low-level database queries (LLDQ) that are needed to gain the requested information. *BlaszeGraph* processes these LLDQ and passes the results to the *Join Results* sub-module. It converts the low-level database results into HLDR. The result is then persisted in the *Storage*, which returns it as the modules output. A detailed illustration of the third module can be seen in Figure 4.4

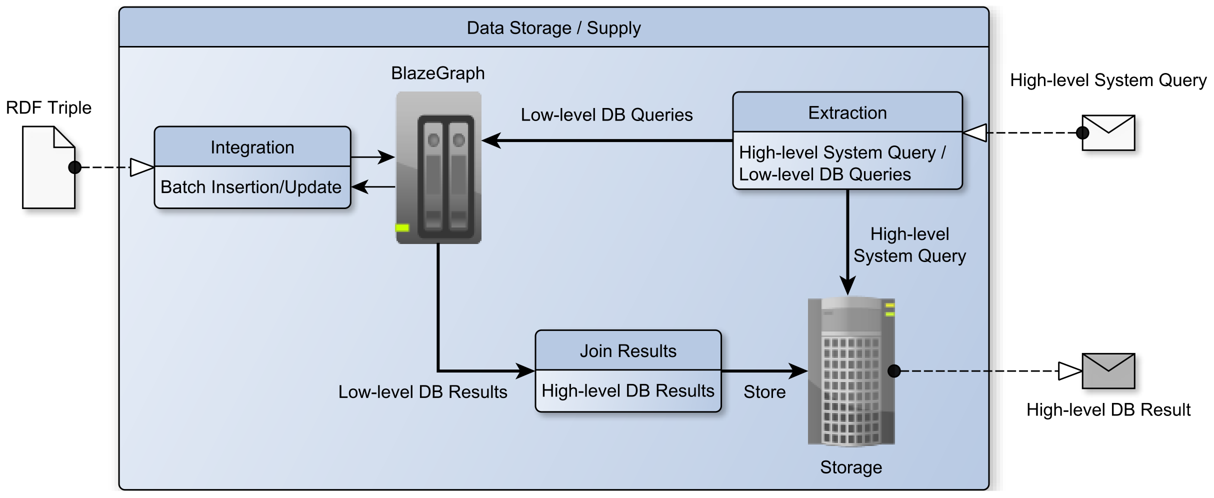

Figure 4.4: Detailed visualization of *Data Storage and Supply* module. It has two different input sources and one output. The first input are the RDF triples that were created during the previous modules. The second input are high-level system queries from the *Gene Ranking System*. The output is a high-level database result that answers the given query.

4.4 Gene Ranking System

The last main module of the entire pipeline is the *Gene Ranking System*. It is divided into two larger sub-modules. The first sub-module is the *Machine Learning* module that handles the user queries and contains all main methods that are used by the system. The second sub-module is the *Evaluation*. It is, technically, not a main part of the system pipeline but it is used for ML-model training, ML-model evaluation and for the evaluation of the final system.

The actual main part of the module is the *Machine Learning* sub-module. A high-level user query (HLUQ) serves as input. The current system can only process a ranking or a classification query. Both queries require the same interaction with the *Data Storage and Supply* module. They do only differ in the used ML-model which is handled by the *Classification and Ranking*-module. Thus, we do not implement a query interpreter but pass the HLUQ directly to the *Feature Extraction* module. It converts the user query into a HLSQ and requests a HLDR from the *Data Storage and Supply* module. The result is used in the *Classification and Ranking* module to create a high-level user result. Therefore, it uses a previously trained ML-model. The output is a high-level user result that contains the requested query. This is either a ranked list of genes for a specific disease (ranking) or a set of disease related genes (classification). The *Evaluation* sub-

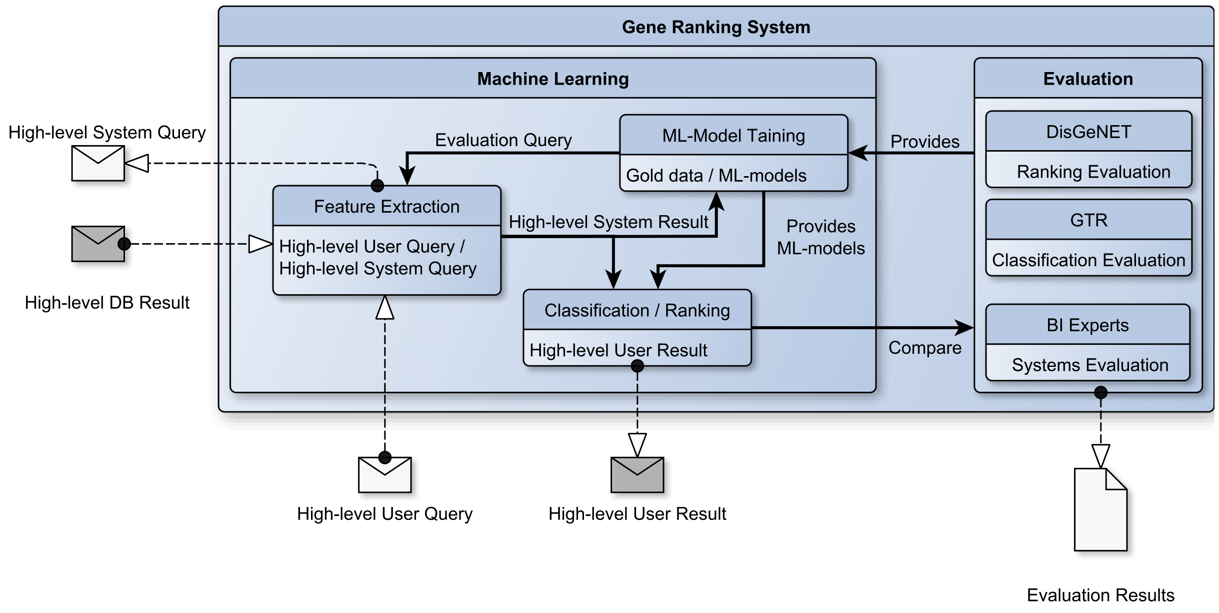

Figure 4.5: Detailed visualization of the *Gene Ranking System* module. It has one user input and one use output (bottom). It shares a bidirectional connection to the *Data Storage and Supply* module (left) and creates evaluation results.

module provides two gold standards, such as *DisGeNET* and *Genetic Testing Registry*. Using these gold standards, two different ML-models are trained in the *Training* module. Thereto it creates evaluation queries with respect to the positive training examples from the gold data. The results are used to train the classification model, respectively the ranking (regression) model. A detailed illustration of the Gene Ranking System module can be found in Figure 4.5.

5 Approach and Resource Generation

This chapter presents the methods of this approach and the resources that are generated during development. The first Section (5.1) is about the disease recognition and the normalization of the extracted disease mentions. We address the task of disease recognition by using the CRF-tagger that was developed by Klinger et al. in [36]. In Section 5.1.2 we introduce *DiNo*: a simple disease normalization framework. Section 5.2 describes the gene-event recognition, followed by the gene normalization and gene filtering. We use TEES [9] for the task of gene-event extraction which subsumed the task of gene recognition. In Section 5.2.2 we explain the integration of the gene normalization framework *GeNo* [72] into the system's pipeline. Section 5.3 describes the generation of the gene-interaction network. Thereto, we introduce an event simplification method and we define a key-term of this thesis, namely *Path-Signatures*. The section ends with the description of the construction of these Path Signatures. In Section 5.4, we discuss the persistent information storage that provides a feature extraction at query time. Those features, that we use for our machine learning models, are formalized in Section 5.5. The chapter ends with a brief summary of the presented methods, the generated resources and the implemented features.

5.1 Disease Recognition and Normalization

This section describes the frameworks that we use to tackle the problem of disease recognition and disease surface form normalization. The first subsection describes briefly the CRF-tagger by Klinger et al. [36]. In Section 5.1.2 we introduce *DiNo*, a simple disease-surface-form normalization framework based on the concept of the state-of-the-art system DNorm [38].

5.1.1 Disease Mention Recognition

We address the problem of disease name recognition by using a natural language tagging framework which is based on conditional random fields. The framework was introduced by Klinger et al in [36]. The tagging method makes use of two different annotation strategies for disease recognition. For each strategy a CRF-model was trained separately. Following a complex pipeline of natural language processing tools for information extraction and sentence tagging, these strategies were applied in order to train two conditional random field-models. During disease recognition on new sentences, these two models were combined to get the best results.

The tagger we use in our approach was trained and evaluated on the Arizona Disease Corpus [40] in cooperation with Matthias Hartung. The evaluation yields a performance of 80.60% for the F-Score, as shown in Table 5.1.

Setup	Precision	Recall	F-Measure
CRF tagger on Arizona Disease Corpus	83.09	78.26	80.60

Table 5.1: Evaluation results for the CRF-tagger on the Arizona Disease Corpus.

We applied this model on the MEDLINE-data presented in Section 3.1 and were able to extract more than 1.5 million instantiations of disease surface forms representing disease candidates. By investigating the extracted disease surface forms, it turned out that the data is very redundant and contains way more surface forms than actual disease types. This leads us to the disease surface form normalization (or abbreviated: disease normalization). We tackle this problem by using *DiNo*.

5.1.2 Disease Surface Form Normalization using DiNo

Implementing a biomedical normalization tool is usually not a simple task and needs a thorough investigation of the considered data. For most biomedical entities, there is already existing literature that describes highly optimized approaches. However, there is always a gap between existence and usability. An integration of a state-of-the-art system into a running pipeline often needs a cooperation with the developer and/or is very time consuming. Due to time issues, our attempt of integrating Robert Leaman's DNorm [38]-approach failed. However, we were able to develop and implement DiNo, a simple disease normalization framework. DiNo follows the example of DNorm which is roughly based on

TF-IDF. It is optimized to the recognized disease mentions that are produced by the CRF tagger. In an evaluation of DiNo, we were able to show that this approach outperforms a previously made attempt as well as our baseline. Hereinafter, both, the implementation of DiNo and its evaluation are described.

DiNo: A simple Disease Normalization Framework The basic idea of DiNo is to calculate a set of scores for a disease mention, for each disease within a comprehensive disease dictionary. The score is a product of the disease name corresponding TF-IDF value and a string similarity function. The disease, to which the highest score was assigned, is determined as the normalized disease. The framework can be decomposed into three sub-components:

1. The generation of a comprehensive disease dictionary.
2. The offline calculation of all TF-IDF values.
3. The online calculation of the set of scores for a disease candidate.

1. Generation of a comprehensive disease dictionary
The first task is the generation of a comprehensive disease dictionary. Each entry in this dictionary contains a preferred disease name, a list of synonyms and a corresponding ID. For this purpose, the two databases MeSH and OMIM are used. The composition of both databases was already described in Section 3.2.

MeSH for disease dictionary MeSH is an ontology based database that supports a hierarchical structure of many biomedical concepts. Each record has a unique MeSH-ID which corresponds to each concept in the specific record. This hierarchical structure of a record is illustrated in Figure 5.1.

The figure shows a simplified hierarchical excerpt of a MeSH descriptor record. Each record has a list of concepts. To each concept, one or more semantic types are assigned. In this example, both concepts are settled in the same semantic type, namely *Disease or Syndrome*. Besides the semantic type, a concept has a list of terms. Each term is a synonym describing the main concept (in this case a disease or a syndrome) of this record. Each dictionary entry has a preferred disease name and a list of synonyms. This preferred disease name can be found in the preferred term of the preferred concept[1].

[1]It can be found by following the Y-marked edges starting at the *Concept List*-node.

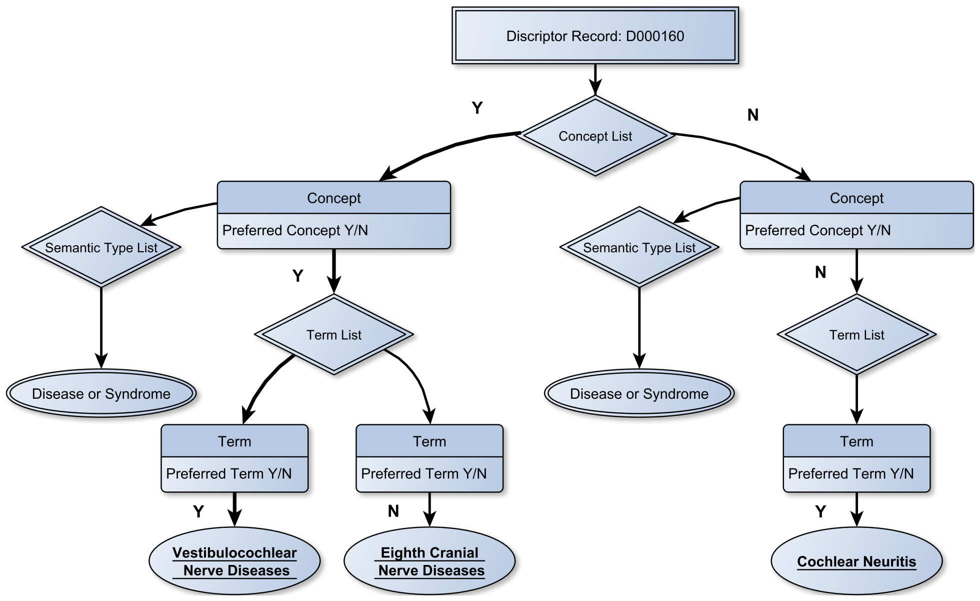

Figure 5.1: Simplified hierarchical representation of a record in MeSH. Note, that this illustration is simplified in the existent information, as well as in the number of concepts and terms.

In the example excerpt in Figure 5.1, the preferred disease name is *Vestibulocochlear Nerve Diseases*. The unique ID can be found in the root-node of the record. It is D000160.

MeSH provides more than 130 different semantical types. However, the records are not restricted to diseases, but include any medical subject. Thus, there are a lot of semantical types that we are not interested in. In fact, as described by Chun in [17], only 12 types may be associated with a disease. We follow the example of Chun and consider all concepts whose semantic type conforms to one of the 12 types that are shown Table 5.2.

OMIM for disease dictionary OMIM is separated into several files. We use the *morbidmap*-file which contains all human diseases that are mentioned in OMIM. The structure of morbidmap is very simple. Each line contains a disease name and a corresponding OMIM-ID. However, some entries are marked with special characters. In the current system we ignore all entries that start with one of the following characters: *?*, *[* or {.

SemanticTypeName	Count
Acquired Abnormality	142
Anatomical Abnormality	143
Cell or Molecular Dysfunction	282
Congenital Abnormality	410
Disease or Syndrome	11.737
Experimental Model of Disease	49
Finding	393
Injury or Poisoning	416
Mental or Behavioral Dysfunction	486
Neoplastic Process	1.179
Pathologic Function	425
Sign or Symptom	720
Total	16.391

Table 5.2: This table contains all semantic types that can be associated with a disease by Chun. We count the number of occurrences and display them in the right column. Since a concept can have multiple semantic types, the total number presented here, is actually higher than the number of considered concepts.

The dictionary The dictionary is built with all entries of MeSH whose semantic type matches with one of those in Table 5.2, and all non-marked entries in OMIM's morbidmap. During the construction of the dictionary, we noticed up to 1.140 overlapping disease names. An example is the genetic disorder *Schwannomatosis*. It consists in OMIM with the ID 162091 as well as in MeSH with the ID C536641. We decided to lay a higher priority to the MeSH entries and assign the MeSH-ID in such cases. The reason for this is the hierarchical structure of MeSH and the supply of synonyms.

Before these disease names are written into a comprehensive dictionary, they are pre-normalized with the following three normalization steps:

1. Each disease name is converted into lowercase.
2. From each disease name, we remove all stopwords (*on*, *in*, *to*, *by*, *at*, *of*, *the*), punctuations and attachments (this includes modifiers like *'s*)
3. Each disease name is tokenized by whitespace-characters, which are used to build a set of tokens.

The pre-normalization has two important reasons. At first, we reduce the amount of entries in the dictionary by deleting semantically redundant disease names. The second reason is to make string matching techniques more accurate.

Table 5.3 shows an example of the *Parkinson Disease(D010300)* record. The first column contains the actual concept term as it is mentioned in MeSH. We marked the preferred disease name of the record with (*). After the normalization, each previous concept-term is a set of tokens. This is displayed in the mid column in set-notation. The last column contains the actual dictionary entry. We collapsed equal sets to one single set. These generated dictionary sets are finally used as synonyms to the preferred set and the corresponding ID. We do the same normalization steps for each OMIM entry and each disease surface form from the two gold-standards.

Concept Term	Normalized Form (Token Set)	Dictionary Form
Parkinson Disease (*)	{parkinson, disease}	{parkinson, disease }(*)
Parkinson's Disease	{parkinson, disease}	
Idiopathic Parkinson's Disease	{idiopathic, parkinson, disease}	{idiopathic, parkinson, disease}
Parkinson Disease, Idiopathic	{parkinson, disease, idiopathic}	
Idiopathic Parkinso Disease	{idiopathic, parkinson, disease}	
Parkinson's Disease, Idiopathic	{parkinson, disease, idiopathic}	
Lewy Body Parkinson Disease	{lewy, body, parkinson, disease}	{lewy, body, parkinson, disease}
Lewy Body Parkinson's Disease	{lewy, body, parkinson, disease}	
Parkinson's Disease, Lewy Body	{parkinson, disease, lewy, body}	
Primary Parkinsonism	{primary, parkinsonism}	{primary, parkinsonism}

Table 5.3: Summary table for the normalization of terms for the *Parkinson Disease*-record. Each entry points at the same disease ID (*D010300*)

The composition, and the total number of diseases that are used to build the dictionary are shown in Table 5.4. Finally, the dictionary consists of 47.336 unique term-sets, that were built out of 73.497 disease names. These unique term-sets are mapped to 17.970 diseases.

	MesH	OMIM	Gold	Total
Disease Names	67.586	3.431	3.138	73.497
Unique IDs	11.401	3.431	3.138	14.832

Table 5.4: Composition of the disease dictionary that is used in *DiNo*.

2. Offline calculation of the TF-IDF values

The second step is to calculate the TF-IDF value for each entry in the dictionary. This requires a definition of the *term*-variable and the *document*-variable. For simplicity, we use in the following description the disease names, but always refer to the corresponding pre-normalized set in the dictionary. We define a document as a set of all disease names and synonyms within the dictionary that points at the same disease-ID. A term is then defined as a single token within the document. Given a term t and a document d the TF value is calculated in formula (5.1):

$$\begin{aligned} \mathrm{tf}(t,d) &= \sum_{t'}^{d} \mathrm{f}(t,t') \\ \mathrm{f}(t,t') &= \begin{cases} 1 & if\ t = t' \\ 0 & else \end{cases} \end{aligned} \tag{5.1}$$

The corresponding term frequency values for the example from Table 5.3 are shown in Table 5.5.

	parkinson	disease	idiopathic	lewy	body	primary	parkinsonism
TF	3	3	1	1	1	1	1

Table 5.5: Term frequency values for all terms in the *Parkinson Disease (D010300)* - document of Table 5.3.

The IDF value is calculated with the standard formula: $idf_t = \log(\frac{|N|}{n_t})$ where N is the number of documents D, defined as $N = |D|$ (number of IDs) and n_t is the number of documents that contain the specific term t. We normalize the TF-IDF value with respect to the sum of lengths of each disease name within the current document. This normalization has the effect that the sum of all TF-IDF values for a specific disease is equal to one, regardless of the number of tokens in the disease name. So, we avoid a bias towards longer disease names.

Consider $T(t,d)$ as a set of all terms in d. The final formula for the TF-IDF value is then presented in Equation (5.2).

$$\mathrm{tfidf}(t,d,D) = \frac{\mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)}{\sum_{t'}^{T(t,d)} \mathrm{tf}(t',d) \cdot \mathrm{idf}(t',D)} \tag{5.2}$$

All these values can be calculated offline.

3. Online calculation of the scores

The next step is the calculation of the final score for each disease name within the dictionary. This is calculated online as described in the following. The determination of the score needs two parameters: a disease candidate in pre-normalized surface form C (as we did in Table 5.3) and a disease name N from the dictionary. In those representations, the two names can be compared in order to create a score.

The score is a composition of two values. The first value is calculated by Equation (5.3). Thereto, we define the similarity function $lsd(t_i^C, t_j^N)$. It returns a maximum similarity score with respect to the Levenshtein distance between $t_i^C \in C$ and a term $t_j^N \in N$. $levenshtein(x, y)$ is defined as the character-based distance between the two tokens x and y as described in [42].

$$mv(t_j^N) = max\{lsd(t_0^C, t_j^N), lsd(t_1^C, t_j^N), \ldots, lsd(t_{|C|-1}^C, t_j^N)\}$$
$$lsd(t_i^C, t_j^N) = \begin{cases} 3 & if \text{ levenshtein}(t_i^C, t_j^N) = 0 \\ 0.9 & if \text{ levenshtein}(t_i^C, t_j^N) = 1 \\ max(0, (0.6 - \sqrt{tfidf_{t_j^N}})) & if \text{ levenshtein}(t_i^C, t_j^N) = 2 \\ -0.1 & else \end{cases} \tag{5.3}$$

We split the string similarity function into four different ranges. If the Levenshtein-distance is equal to zero, the two terms are equal as well. In this case the function returns the maximum score of three. We assume that a Levenshtein-distance of one indicates a spelling error or a spelling variation (e.g. *tumour* instead of *tumor*). In this case, the function returns a similarity score of 0.9. A distance of two may have different meanings. On the one hand, it could be a spelling error or spelling variation in longer tokens. On the other hand, it could have a complete different semantic meaning as in *tremor* and *tumor*. An analysis of the considered disease candidates shows that the latter case often co-appears with a very high TF-IDF score. Therefore, the return value for a distance of two depends on the TF-IDF score of the dictionary-term. We denote it by $tfidf_{t_j^N}$. If the distance is larger than two we assume that the two terms are always semantically different. In this case we return a small penalty score of -0.1. The final return value is the maximum of all compared term pairs.

Up to this point, we compared two disease tokens with focus on the dictionary entry. In the following, another and even stronger penalty function is defined which lays the focus on the disease candidate. For each term in C that is not in N, a penalty value is returned.

This penalty is defined with respect to the total number of terms in C. The formula is illustrated in (5.4):

$$pen(t_i^C) = \begin{cases} -\frac{2.5}{|C|} & if\ max\{lsd(t_i^C, t_j^N) : j < |N|\} > 0 \\ 0 & else \end{cases} \tag{5.4}$$

With these two functions, the final score can be calculated. This is done by Equation (5.5).

$$DiNo\text{-score}(C, N) = \sum_{i}^{|C|-1} pen(t_i^C) + \sum_{j=0}^{|N|-1} mv(t_j^N) * tfidf_{t_j^N} \tag{5.5}$$

The entry in the dictionary with the highest score is supposed to be the normalized form for the disease candidate. The normalized form D_{norm} is finally determined by (5.6).

$$D_{norm}(C) = max\{DiNo\text{-score}(C, N)\ :\ N\ \in\ Dictionary\ \} \tag{5.6}$$

Real World Example for dysfunction on lipid metabolism In the following, we give a real world example for the disease mention *dysfunction on lipid metabolism.* The disease surface form is compared to two different entries in the dictionary. Thereto, we define $C = \{dysfunction, lipid, metabolism\}$, $N_1 = \{lipid, metabolism, disorders\}$, and $N_2 = \{lipid, metabolism, inborn, errors\}$. Notice that, due to the pre-normalization, C does not contain the stopword *on* anymore. Table 5.6 shows the comparison of C and N_1. Table 5.7 compares C with N_2.

t_i^C	pen(t_i^C)	Value
dysfunction	-2.5 / 3	-0.833
lipid	0	0
metabolism	0	0
(Penalty) Score =		**-0.833**

t_j^N	tfidf$_{t_j^N}$	mv(t_N^j)	Value
lipid	0.467	3	1.422
metabolism	0.392	3	1.195
disorders	0.141	-0.1	-0.014
Score =			**2.603**

Table 5.6: Comparison of the disease mention *dysfunction on lipid metabolism* and the dictionary disease *lipid metabolism disorders.*

The penalty score is calculated on the left of Table 5.6. Since there is only one term in C that is not in N_1, namely *dysfunction*, and $|C| = 3$, the penalty is set to $-2.5/3$. On the right side of Table 5.6, the positive (maximum) score is calculated. There are two terms that match exactly to one term in N_1 (*lipid* and *metabolism*). The TF-IDF values for both terms will be multiplied with three. However, *disorders* cannot be found in C, so that the mv(*disorders*) adds a small penalty of -0.014. This penalty has very little effect, due to the very low TF-IDF value of the frequent term *disorders.* Table 5.7 shows the score calculation of N_2.

t_i^C	pen(t_i^C)	Value
dysfunction	-2.5 / 3	-0.833
lipid	0	0
metabolism	0	0
(Penalty) Score =		**-0.833**

t_j^N	$\text{tfidf}_{t_j^N}$	$\text{mv}(t_N^j)$	Value
lipid	0.307	3	0.921
metabolism	0.258	3	0.774
inborn	0.311	-0.1	-0.031
errors	0.132	-0.1	-0.013
Score =			**1.651**

Table 5.7: Comparison of the disease mention *dysfunction on lipid metabolism* and the dictionary disease *lipid metabolism inborn errors.*

Table 5.7 can be interpreted analogously to Table 5.6. However, the positive (maximum) score is with 1.651 one point lower. This has several reasons. First, N_2 is one term longer that N_1. Thus, the TF-IDF values for *lipid* and *metabolism* are both lower[2]. On top of that, N_2 contains a third strong term, namely *inborn*, which is not in C.

In Equation (5.7), the values for both dictionary entries are calculated. Equally to this, the score for each entry in the dictionary is calculated. By considering only these two entries, the disease mention C would finally be mapped to N_1.

$$\begin{aligned} DiNo\text{-score}(C, N_1) &= -0833 + 2.603 = 1.77 \\ DiNo\text{-score}(C, N_2) &= -0833 + 1.651 = 0.818 \end{aligned} \tag{5.7}$$

Evaluation of the DiNo Framework We evaluated DiNo on the NCBI disease corpus and compared it against three different approaches. The first algorithm is a simple normalization approach, which normalizes disease candidates by using the pre-normalization process that was described in Table 5.3. This approach, hereinafter referred to as SDN, is our first baseline. The second approach normalizes disease candidates by using the framework of Paassen [50]. We generated the ontology based database by mapping the structure of MeSH and extended them by the considered OMIM entries. This approach is called ODN and serves as our second baseline. On top of that, we compare DiNo against the state-of-the-art framework *DNorm* as described in the literature in [38].

[2]The sum of all TF-IDF values within a dictionary entry is equal to one.

The settings of the evaluation are equal to those described in Robert Leaman's evaluation in [38]. We only compare the generated sets of normalized disease names for a given abstract. We ignore the exact locations and the total number of diseases. The number of true positives is thereby defined as the size of the intersection between the gold set of diseases and our generated set. We use the measures precision, recall and harmonic F-Score to calculate the micro-average results.
During development, we adjust four parameters to optimize DiNo. This includes all return values of the lsd-function that were shown in Equation (5.3), as well as the penalty parameter in Equation (5.4). We found out that the presented values lead to a stable result. These parameters were set during development and also used in the final evaluation. The disease mention recognition was done with the previously described CRF tagger from Section 5.1.1, since DiNo and the two baselines are restricted to normalizations. The performances of all approaches are displayed in Table 5.8. The approach of the first baseline

Setting	Precision	Recall	F-Score
SDN	50.16	26.03	34.28
ODN	41.21	36.21	38.55
DiNo	61.56	61.42	61.49
DNorm (From Literature)	80.30	76.30	78.20

Table 5.8: Micro average results for four different disease normalization approaches on the NCBI disease corpus test set.

contains a simple cutoff after the first comma and convert it into lowercase. The relative high precision of 50.16% is probably largely influenced by the high performance of the CRF tagger (80.60% in F-Score). Although the CRF tagger does not normalize the found disease names, it is very likely that some of them are already in their preferred form and matches exactly a disease name in MeSH or OMIM. However, this simple normalization has obviously a negative effect on the recall which is at 26.03%. A more balancing approach states the second baseline. It has a lower precision but a higher recall than the first baseline. However, the F-Score is just slightly higher (38.55%). We were able to outperform the two defined baselines with DiNo. Both, the precision and the recall are significantly higher with 61.56% in precision, respectively 61.42% in recall. That leads to an F-Score of 61.49%. By implementing this relatively simple approach, we outperform both baselines by approximately 25 points. Unfortunately, we cannot range on the same level as the highly improved state-of-the-art approach DNorm.

5.2 Extraction of Gene Interaction Events and Gene Normalization

In the previous section we described the process of disease recognition and normalization. We successfully applied the CRF tagger in combination with DiNo to all MEDLINE abstracts and were able to extract more than 12 thousand diseases. The next step is to extract and normalize all genes from MEDLINE. This section is about the gene recognition, normalization and gene event extraction. The task of gene recognition is subsumed by the gene event extraction, since we are mainly interested in those genes that are involved in an event. Further, we only consider protein-coding genes[3], we assume semantical equality between a gene and its protein. Thus, we do not distinguish between a gene and a protein. We extract gene-events and thereby all involved genes by integrating the state-of-the-art NLP tool TEES [9]. In a subsequent process, all found gene candidates are normalized with GeNo [72] and filtered for human genes by comparing the taxonomic ID with EntrezGene. We evaluate this pipeline against two baselines on the BioCreative-II gene normalization test set.

5.2.1 Gene Event Recognition

We integrated the natural language processing tool called Turku Event Extraction System, which is a state-of-the-art framework in the biomedical domain that extracts complex gene-events on sentence basis. The pipeline of TEES consists of several NLP methods to convert the plain textual input into a TEES readable format. For this purpose the GENIA sentence splitter [55] is applied to tokenize the plain text. Each sentence is then parsed with the Stanford sentence parser [16] to structure them. In the next step, the NER-system BANNER [39] is used for gene recognition. The output of these consecutive pre-processings are comprehensively tagged sentences.

The actual event extraction pipeline of TEES starts with a trigger recognition for possible events. A trigger is defined as a word that probably states a relation between two genes e.g. *induced*. The previously generated parse tree of a sentence is then extended by the trigger annotations. The parsed sentence is used in combination with the annotated triggers to create a semantical representation of the sentence. In this step, genes and triggers were linked. Such a linking is called edge. After the detection of all edges, they are converted into events. An event may consist of several edges. The complexity of a

[3]We use EntrezGene to identify protein-coding genes.

single event is thereby defined through the number of edges, respectively the number of considered genes and triggers.

The data structure of TEES An example of a fully annotated complex sentence is illustrated in Figure 5.2.

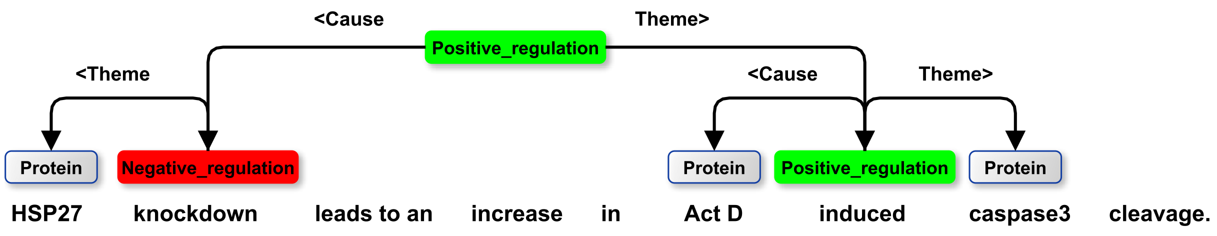

Figure 5.2: The visualization shows a fully parsed and annotated example sentence.

TEES was developed to challenge the GENIA event extraction task [33]. Thereby, the structure of the extracted information is consistent to the event scheme and the output format that was especially designed for that task. This scheme is illustrated in Table 5.9.

The left column lists all possible event types that can be triggered. The first six event types are restricted to a single core argument (a protein). However, the regulation-events can have either one or two core arguments. These core arguments are placed in the mid column. They differ in *Theme* and *Cause*. As we can see in Figure 5.2 both, *Theme* and *Cause* are semantic labels, describing a relation. The actual core argument is either a protein or another event type. The right column contains additional information that can be extracted, if any, to each protein.

Event Type	Core arguments	Additional arguments
Gene expression	Theme(Protein)	
Transcription	Theme(Protein)	
Protein catabolism	Theme(Protein)	
Localization	Theme(Protein)	AtLoc(Entity), ToLoc(Entity)
Binding	Theme(Protein)	Site(Entity) +
Phosphorylation	Theme(Protein)	Site(Entity)
Regulation	Theme(Protein / Event), Cause(Protein / Event)	Site(Entity), CSite(Entity)
Positive regulation	Theme(Protein / Event), Cause(Protein / Event)	Site(Entity), CSite(Entity)
Negative regulation	Theme(Protein / Event), Cause(Protein / Event)	Site(Entity), CSite(Entity)

Table 5.9: GENIA Event Scheme that is used by TEES.

TEES generates two files that contain the extracted information in a structured format. A detailed description of the output format can be found on their website[4].

TEES was successfully evaluated on the BioNLP 09 Shared Task [8], the BioNLP 11 Shared Task [10] and on the drug-drug interaction challenge [7] as displayed in Table 5.10.

Task	F-Measure	Rank
BioNLP 09	52.86	1st place
BioNLP 11	51.95	1st place in 4/8 tasks
DDC 11	62.99	4th place

Table 5.10: Evaluation of TEES in several tasks and their ranking.

By integrating TEES, we can extract all gene-events from a given input source and thereby all genes that are actually involved in an event. However, TEES does not contain a built-in gene normalization tool. Likewise to the diseases, all genes need to be normalized in a subsequent process after their recognition. We normalize all gene mentions by using a state-of-the-art normalization framework called GeNo [72]. This process is described in the following section.

5.2.2 Gene Surface Form Normalization

In the previous section we described the Turku Event Extraction System that we use in our approach for gene recognition and gene-event extraction. These gene mentions are in textual surface form and need to be normalized and disambiguated. This is important to increases the data density, it enables standardized queries and facilitate the uniqueness of the systems output. Due to the high variability in gene surface forms, a normalization is both, an extensive and important task. The variation of the gene surface forms is mostly given through the high number of synonyms, abbreviations and species depending spellings. Thus, a gene normalization cannot be done with a single dictionary based approach, but requires contextual informations. To tackle the problem of gene normalization, we use GeNo [72].

[4]`http://2011.bionlp-st.org/home/file-formats` Access: 2015-04-22

GeNo is a highly performing system for gene name recognition and normalization by employing multiple statistical and symbolic methods. GeNo's pipeline can be roughly organized into four steps. The pipeline starts with the building of a comprehensive gene dictionary. It includes data from curated gene databases like EntrezGene [45] and UniProt [18]. The dictionary is augmented by creating variations for each entry. These variation generator uses approved naming rules and tokenization procedures to convert a gene surface form to its normalized form. The second step is the gene recognition and normalization. For this purpose, the machine learning approach Jnet [21] is integrated. To increase the recall, a dictionary-based method is applied, too. The normalization of gene candidates is done with the same method that was used for the variant generation for curated gene names, as described earlier. The last step is the gene mapping component, which uses a string matching technique to compare each candidate with each entry in the dictionary. As mentioned in [72], string matching techniques rather lead to a high recall instead of optimizing the precision. To decrease the number of false positives, a semantic similarity scorer is applied to each gene candidate. If this similarity score is below a specified threshold, the candidate is discarded. The result is a list of genes with a corresponding confidence score. The higher the score, the more likely it is that the given gene candidate is related to the corresponding dictionary entry.

The text above describes briefly the main procedure of GeNo. However, as mentioned before, we use TEES for gene recognition. That forces us to modify GeNo's original pipeline. In fact, we replace their gene recognition module (Jnet and the dicitonary-based approach) with TEES. All gene mentions that are found by TEES are converted into a pre-normalized form like before. After that, the pipeline continues as usual.

Evaluation of the Gene Normalization In the following, the evaluation result of the modified GeNo pipeline is described, hereinafter referred to as (TEES+GeNo). We use the pre-trained model *GE11* for TEES which was specially trained for the task of gene-event recognition within plain textual abstracts and was evaluated on the BioCreativ-II event recognition task. We evaluate (TEES+GeNo) against two baselines on the BioCreative-II gene normalization test set [46] and compare our results to the original GeNo pipeline (GeNo) that was described in [72]. Both baselines use TEES in the same way as previously described for the task of gene recognition.

Similar to the disease normalization evaluation, the first baseline is a very naive approach. Gene names are case sensitive and often contain special characters like hyphens and apostrophes. Thus, the first baseline approach simply cuts each gene candidate after the first comma. This is motivated through observations that the cut part often contains additional information or gene-specifications. We name this approach (TEES + SGN).

The second baseline uses the ontology based approach by Paassen [50], that was also used for the disease normalization. To build the ontology database all human genes that are mentioned in EntrezGene are considered. Since EntrezGene is not hierarchically structured, each gene that is labeled with the *protein-coding* tag represents a single root node in the database. In order to simulate the hierarchical structure, each synonym[5] for a gene serves as child node. This second baseline is named (TEES + OBGN).

We evaluated all previously mentioned gene normalization approaches on the BioCreative-II gene normalization task [46]. It challenges researchers to build an approach for gene recognition and/or gene normalization. The task is specified to human-genes and does not take other species into account. The test-set consists of 262 PubMed-abstracts with 656 annotated human-genes in total. Each annotated gene includes information about the surface form, an EntrezGene-ID and a PubMed-ID. The actual position within the abstract is neither given nor demanded.

The performances of the tested approaches are shown in Table 5.11.

Setting	Precision	Recall	F-Score
TEES + SGN (first baseline)	13.34	5.87	8.15
TEES + OBGN (second baseline)	28.37	16.39	20.78
TEES + GeNo	56.33	54.37	55.33
GeNo (From Literature)	87.8	85.0	86.4

Table 5.11: Evaluation results for several gene normalization approaches on the BioCreative-II gene normalization test set for human genes.

The first baseline has a very low F-score of 8.15%. The result is composed of a precision of 13.34% and a recall of 5.87%. The low result suggests a high complexity of the general task of gene recognition and normalization. The first baseline covers just a simple dictionary lookup without any variant generation or extensive normalization. Thereby the low

[5]We use the following EntrezGene fields as synonym: *Synonyms*, *Symbol_from_nomenclature_authority*, *Full_name_from_nomenclature_authority*, and *Other_designations*

precision is probably an effect of the huge gene name variation that can be observed in the data. The main cause of the low recall is probably the very strict threshold of a pure lookup with a zero error rate for string matching.

The result of the second baseline shows a slightly better performance. Both, precision and recall are higher than the first baseline. The result is an F-score of 20.78%. The increasing performance could be an effect of a general variant generation and normalization that is used in this approach. However, neither the variant generation nor the generalization is specifically tailored for gene names. This has a negative effect since common gene names differ in their structure to other named entities e.g. disease names. This leads us to the assumption, that a general variant generation and normalization for named entities have little effect to specific and more complex named entities.

The pipeline of GeNo consists of such a customized variant generation and normalization. GeNo is strongly specified to the task of gene normalization. This can be recognized in the evaluation results. Our gene normalization pipeline (TEES + GeNo) outperforms the two baselines with an F-score of 55.33%. Unfortunately, we were not able to reproduce the original GeNo score from the literature. This is most likely the effect of merging two single state-of-the-art systems that are not adapted to each other. In contrast to GeNo, TEES and thereby the gene recognition, is not optimized to human genes, but finds each possible candidate regardless of the taxonomic scope. This could have a negative effect on the precision. The relatively low recall needs further investigations on TEES. A first step to explain the recall could be an evaluation of TEES for pure gene recognition. However, this goes beyond the scope of this work and is left for future work. The evaluation results presented here do not exactly reflect the pipeline that is needed in our approach. To the best of our knowledge, there is no existing benchmark that exactly captures the event-recognition only for human genes. Although we outperform the two given baselines, in future work we will need to work on this issue.

The normalization of biomedical entities is a crucial task of this approach. Since the gene normalization is at the very beginning of the system, it could cause a lot of damage. Wrongly normalized entity names influence the entire system and could lead to bad evaluation results and, with respect to the discovery of public knowledge, to the generation of wrong assumptions. Especially for the task of hidden knowledge discovery and predicting new gene candidates, it is mandatory to make the process of normalization re-traceable. This helps researchers and annotators to understand the systems output more effectively.

5.3 Gene-Interaction Network

In this section, we describe the generation process of a comprehensive gene-interaction network. Thereto, we discuss the gene-event normalization/simplification and the process of event graph transformation. An event has been defined as a set of coherent event types, genes and additional arguments. In most cases, an event consists of more than one event-type. Such complex events need to be simplified in a graph transformation process. The objective of the transformation is to convert a given arbitrarily complex event into simpler sub-events. We do this in order to reduce the complexity of the data and make them persistently storable into our database. As mentioned before, TEES generates plain textual output following the file description of the event extraction task. These files contain structured information about the graph structure for each gene-event that was extracted. The first step that needs to be done, is to convert this textual output into a graph structure. Since our entire system was written in the programming language java [5], we map the input into a java tree object. After that, the internal tree representation is transformed into simpler sub-events. Therefore, we extract several subtrees. Each sub-tree is restricted to a maximum number of two leaf-nodes, respectively two genes, describing a simplified event between both. We call the converting from text into a tree *Graph Construction* and the latter step *Graph Transformation.*

5.3.1 Graph Construction and Transformation

In order to make the graph transformation process more comprehensible, we illustrate it based on the example sentence shown below. This example sentence is an excerpt of the publication: "*Upregulation of heat shock protein 27 confers resistance to actinomycin D-induced apoptosis in cancer cells.*" with the PubMed-ID 23848600.

(5.8) ***HSP27 knockdown*** *leads to an* ***increase*** *in* ***Act D-induced caspase 3*** *and caspase 7 cleavage, and sensitizes rhabdosarcoma cells and breast cancer cells to Act D-induced apoptosis[...].*

Graph Construction In Section 5.2.1, two files were mentioned that are generated by TEES. Usually, these files contain several hundred lines each. The complexity depends on the number of events, proteins and additional information that was found in the input text. In a merging process, all event depending lines from both files are extracted. An example of such extracted information from the sentence in Example 5.8 is shown in Listing 5.1.

The generated file can be roughly divided into three different types of lines. Line 1, 2 and 3 belong to a protein/gene[6]. In this example, there are three different proteins *HSP27*, *Act D* and *caspase 3*. The next three lines are specifications for a (sub)-event and contain meta information. The last three lines represents the actual events. Line 7 shows an event that is bounded to a single protein. The *Negative_regulation* (E195) is bounded to *HSP27* over a *Theme*-identifier. There is no specification of how this regulation needs to be initiated. Line 8 shows a very frequent event type. This *Positive_regulation* (E196) connects two genes, precisely *Act D* as cause-gene and *caspase 3* as theme-gene. Usually, this is already a valid event on its own. However, it exists a more complex event that subsumed E196. This complex event is shown in the last line. It shows a complex event between two events. The cause-event *E195* causes a positive regulation to the theme-event *E196*. Considering this example, we have in fact two valid events: *E196* and *E198*. Since *E196* is subsumed by *E198*, we keep that in mind but lay the focus in the following to the more complex event.

```
1 T371   23848600     758     763     Protein HSP27
2 T372   23848600     798     803     Protein Act D
3 T373   23848600     812     821     Protein caspase 3
4 T706   23848600     764     773     Negative_regulation     knockdown
5 T707   23848600     804     811     Positive_regulation     induced
6 T708   23848600     786     794     Positive_regulation     increase
7 E195   Negative_regulation:T706     Theme:T371
8 E196   Positive_regulation:T707     Cause:T372    Theme:T373
9 E198   Positive_regulation:T708     Cause:E195    Theme:E196
```

Listing 5.1: Merged information of the sentence in Example 5.8.

Up to this point, we have an understanding of the textual representation of a complex event. The next step is to convert these lines into a graph. To address this problem, we implemented a simple tree object in java, covering all necessary functions to store all needed information. Figure 5.3 shows the graph representation of the complex event *E198*.

[6]As mentioned before, we do not differentiate between genes and proteins.

After converting the textual representation into a graph, the complex event can be transformed into sub-events. By that, we assume that there is a possibility of dividing the event-tree into sub-trees, without any or at least, with an acceptable loss of semantic information. Besides minimizing the loss of information, it is also important to avoid the generation of new and possibly wrong information. Tackling these requirements we developed an algorithm, in agreement with a biomedical expert, that takes both conditions into account.

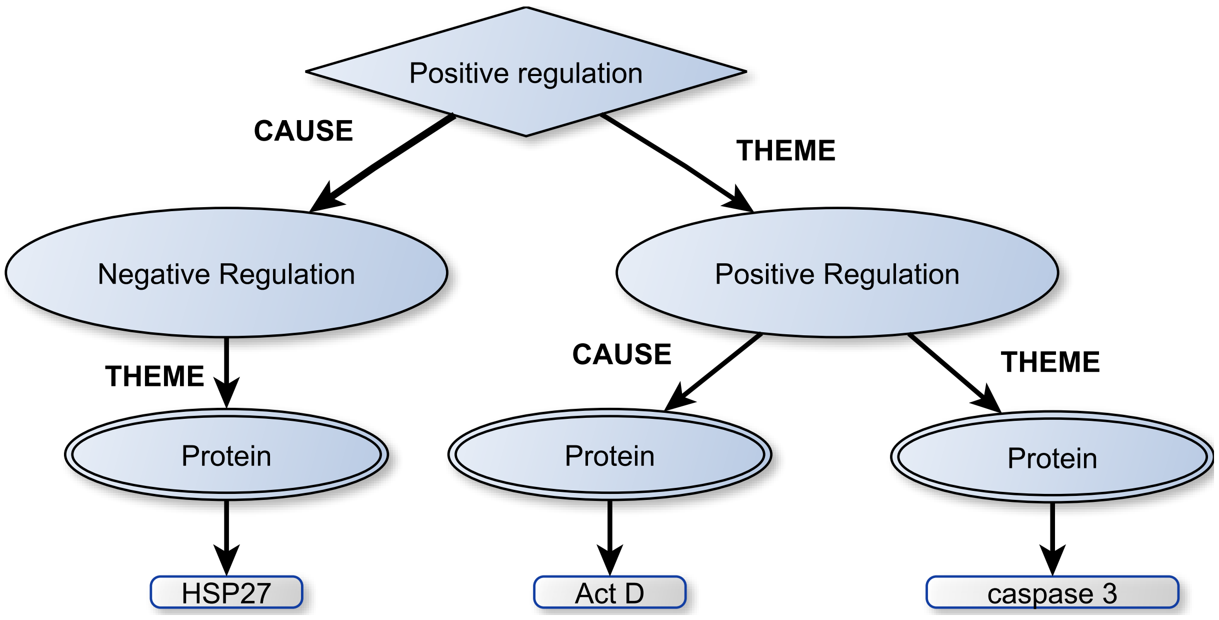

Figure 5.3: Graph representation of the complex event E198 that was shown in Listing 5.1.

Graph Transformation The algorithm is composed of two components that are called subsequently for each complex event-tree. The objective of the graph transformation is to convert complex events into simpler sub-events that can be stored in our database. Thereby, we create a comprehensive gene-interaction network.

1. Determination of the cause-gene and the theme-genes
We define a *cause-gene* as a gene, which manipulated at least one other gene. Such a manipulating role is defined by the semantic keyword *Cause*. Besides that, we define a *theme-gene* as a gene that is manipulated by at least one cause-gene. We assume, that a theme-gene within a cause-event of a complex event, has a manipulating role to each gene in the sub-tree theme of the complex event. Taking the example from Figure 5.3, there

are two valid cause-genes, namely *HSP27* and *Act D*. If the cause is *HSP27*, then the set of themes contains *Act D* and *caspase 3*. If *Act D* is chosen as cause, then *caspase 3* is the only theme. An other example is illustrated in Figure 5.4. In this example event, *gene1* serves as cause for *gene2* and *gene3*. *gene2* serves as cause of *gene3*. However, *gene3* is not a valid cause-gene.After the determination of causes and themes, a list of tuples can be created. Each tuple consists of exactly one cause-gene and one corresponding theme-gene.

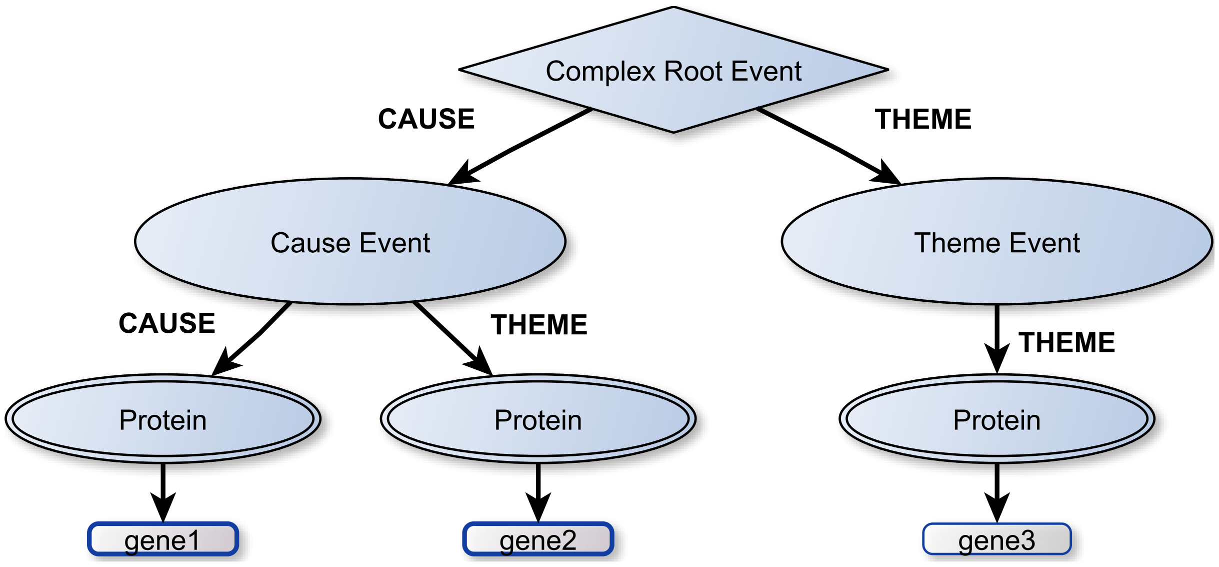

Figure 5.4: Complex event with a valid cause event. Each gene in the cause event serves as cause for the gene in the theme-event. All possible cause-genes are outlined boldly.

2. Generate sub-trees for each tuple

In the second step, we use a simple form of the Dijkstra algorithm [60] to extract the path between the cause and the theme within a tuple. The Dijkstra algorithm always finds the shortest path between two nodes in a given graph. Since each event is represented in a tree rather than a graph, there is just one solution. By using this algorithm, a set of sub-trees is generated, representing each sub-event of a complex event. Each tree within the generated set consists of exactly two leaf-nodes (the cause-gene and the theme-gene) and at least one event-type-node. In addition, two nodes are connected trough exactly one directed edge, outgoing from the cause-gene. Three subtrees can be extracted from the example in Figure 5.3. These are illustrated in Figure 5.5 and 5.6.

At the beginning of this section, we considered the complex event and mentioned a second valid event (E196). By applying the graph transformation to the complex event, every possible valid event is taken into account. However, a last step is required to convert the event trees into the database readable format. In this step, all non-leaf nodes and edges are collapsed into a single edge. This new edge contains each information that was previously represented from all edges and nodes between the two leaf-nodes. Due to the structure of a sub-tree, this conversion can be done without any loss of information. The result can be written as triple, that consists of two genes and their complex relation.

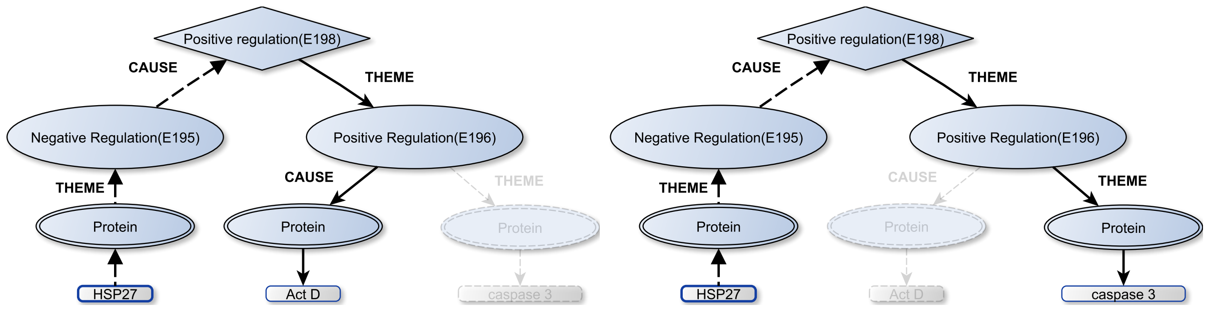
Positive regulation(E198)
CAUSE
THEME
Negative Regulation(E195)
Positive Regulation(E196)
THEME
CAUSE
THEME
Protein
Protein
Protein
HSP27
Act D
caspase 3
Positive regulation(E198)
CAUSE
THEME
Negative Regulation(E195)
Positive Regulation(E196)
THEME
CAUSE
THEME
Protein
Protein
Protein
HSP27
Act D
caspase 3

Figure 5.5: This figure contains two extracted subtrees for the example tree in Figure 5.3. Both of them have *HSP27* as cause. The left figure shows the path to *Act D*, the right figure to *caspase 3*. The edges are adjusted to be outgoing from the cause-gene, marked as dashed lines. However, the edge descriptions do not changed.

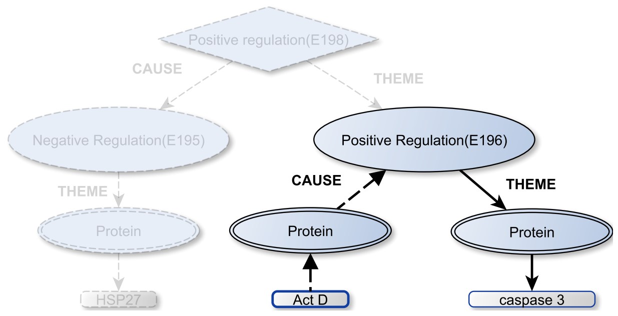
Positive regulation(E198)
CAUSE
THEME
Negative Regulation(E195)
Positive Regulation(E196)
THEME
CAUSE
THEME
Protein
Protein
Protein
HSP27
Act D
caspase 3

Figure 5.6: This figure contains the third extracted sub-tree between *Act D* (cause) and *caspase 3* (theme). The edges are adjusted to be outgoing from the cause-gene, marked with dashed lines. However, the edge descriptions do not changed.

For the three extracted sub-trees in Figure 5.5 and 5.6, we finally generate three following new trees as shown in Figure 5.7. With these triple-like trees, a comprehensive gene-interaction network can be built, using the graph storage ability of our database. The actual storage is described in Section 5.4.

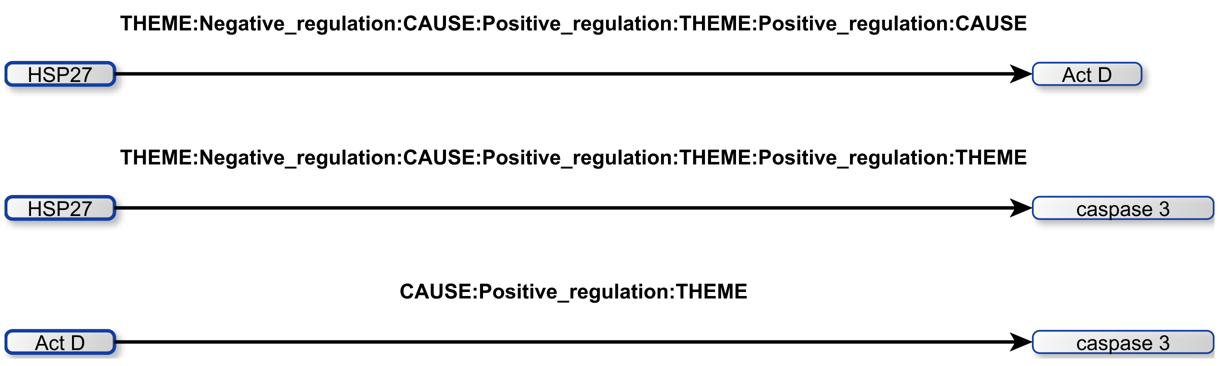
THEME:Negative_regulation:CAUSE:Positive_regulation:THEME:Positive_regulation:CAUSE
HSP27
Act D
THEME:Negative_regulation:CAUSE:Positive_regulation:THEME:Positive_regulation:THEME
HSP27
caspase 3
CAUSE:Positive_regulation:THEME
Act D
caspase 3

Figure 5.7: This figure contains all three final trees that are generated by the graph transformation from the tree in Figure 5.3.

5.3.2 Extraction of Path Signatures

In the previous section, the process of generating the data for a comprehensive gene-interaction network was described. The result was a set of apparently independent triples, that can be stored in the database. However, the database generates a comprehensive network considering all genes and their transformed events. The integration into the database is described in Section 5.4. The interaction network enables the connection of genes beyond their abstract based relation and find new undiscovered public relations [65] between genes. Therefore, we define a new term. We call consecutive relations between two genes in the interaction graph *Path Signature*, referred to as signature. These signatures are later used to formalize some features.

In the following, we describe the extraction of such signatures. On top of that, we introduce a signature simplification/normalization process in order to limit the variance.

Since the network contains directed edges, a distinction between incoming and outgoing signatures is needed. Thus, a signature can be either starting from a cause-gene g_{cause} and ending at a theme-gene g_{theme} or vice versa. Thereby it may have to pass some nodes. These passing nodes are called *bridge-genes*. A signature is defined as outgoing if each edge between g_{cause} and g_{theme} is outgoing from the current node. An incoming signature can be defined analogously. Due to the cyclic structure of the network and the huge amount of data it is also crucial to limit a signature in its length by setting the maximum number of bridge-genes. The generation of a signature from the gene-interaction network is composed of three steps. All three steps are explained with respect to an outgoing signature. Incoming signatures can be generated analogously. The generation of g_{cause} related signatures starts by the extraction of all g_{cause} related raw-signatures.

1) Extraction of Raw-Signatures from Database The first step is to find all paths with a specific length between g_{cause} and g_{theme} in the network. This is handled by the database (see Section 5.4) with the following query:

```
PREFIX e: event
SELECT ?OutRel1, ?OutRel2  WHERE
  {
    <e:HSP27> ?OutRel1 ?BridgeGene1 .
    ?BridgeGene1 ?OutRel2 <e:caspase+3> .
  }
```

Listing 5.2: SPARQL query to retrieve a list of outgoing paths between *HSP27* and *caspase3*. Each path consists of two outgoing relations in consecutive order.

A maximum number of one bridge-gene leads to a number of two relations between these genes. A maximum number of two bridge-genes leads to three relations etc. Figure 5.8 illustrates an example gene-interaction network. It shows two possible outgoing paths starting at *HSP27* and ending at *caspase 3* and passing one bridge-gene. These paths are highlighted as thick lines. The dashed lines are examples of incoming relations, starting at *HSP27* and ending at *caspase 3*. In this example, an incoming path exists only over two bridge-genes. For convenience, all relations are abbreviated with *Rel* followed by a unique ID e.g. *Rel3*. These relations can be very complex as mentioned in Section 5.3.

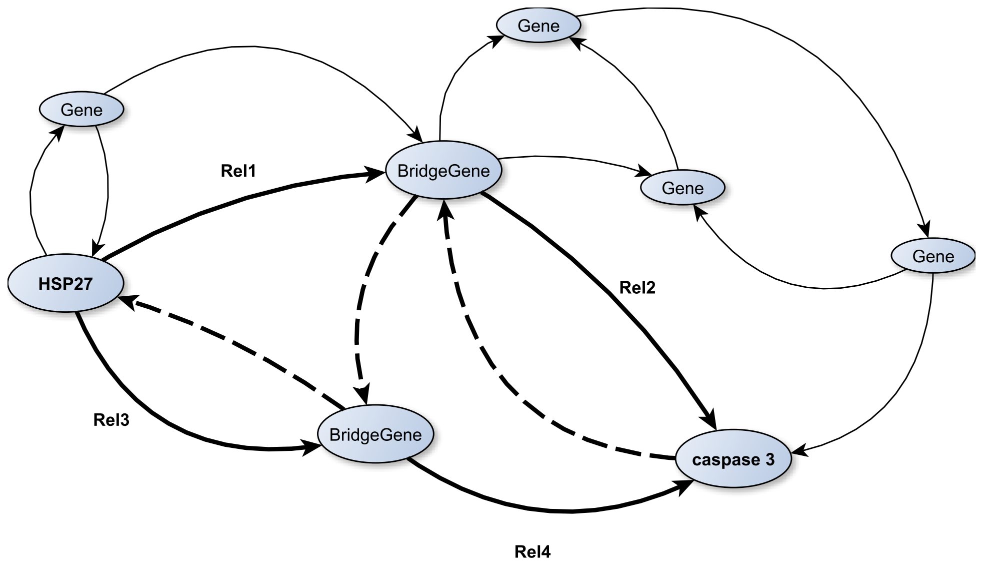

Figure 5.8: Illustration of a gene-interaction network of eight different genes. The relations between *HSP27* and *caspase 3* are emphasized. The thick lines show outgoing relations, starting at *HSP27* and ending at *caspase 3*. The dashed lines illustrate incoming relations starting at *HSP27* ending at *caspase 3*.

The result of this component is a set of lists of relations between g_{cause} and g_{theme}, sorted in consecutive order. Each list contains exactly n relations where n is the number of bridge-genes+1. From the example graph in Figure 5.8, two outgoing relation-lists for *HSP27* can be extracted:

(5.9) (Rel1, Rel2)

(5.10) (Rel3, Rel4)

Due to lack of space, we create the following abbreviation table that is used in the following step.
We create the corresponding raw-signatures by building a comprehensive path, representing all relations in the list. We do this by merging them, separated by colons, in consecutive order as shown in Listing 5.8

Full Term	Abbreviation
CAUSE	CS
THEME	TH
Positive_relation	Pos_rel
Negative_regulation	Neg_rel
Localization	Loc

Figure 5.9: Abbreviation table for event types.

```
TH:Neg_reg:CS:Pos_reg:TH:Pos_reg:TH:CS:Pos_reg:TH
CS:Pos_reg:TH:CS:Neg_reg:TH:CS:Pos_reg:TH:Loc:TH
```

Listing 5.3: Two raw-signatures that were extracted from the example interaction network in 5.8.

2) Signature Normalization In order to reduce the variability and make signatures more generalizable, each relation is normalized in a subsequent process. The complexity of a raw-signature is determined by the event-scheme in Table 5.9 and the number of bridge-genes. Since we want to use the final Path Signature as features, their variability needs to be reduced. This is done by a normalization / generalization of the raw-signatures. A generalization always includes a loss of information. To minimize this loss, we have made two assumptions in collaboration with biologists.

1. Abstraction from Semantic Path Information
The first assumption is, that the current notation contains some terms with very low information content. These are all terms that are actually not an event, in this example *CAUSE* and *THEME*. Both of them can be seen as semantic values that carry information of the original event. If the Path Signature would remain in this notation, there would be a lot of variations that may express the same or almost the same information with a usage of slightly different terms. An example can be seen in Figure 5.7. The first two

relations differ only in the last term (THEME/CAUSE). Assuming that these terms carry only tree depending information, a crucial step is to remove all terms that do not belong to an actual event. The modified forms so far, are listed in Listing 5.3.

```
Negative_Regulation : Positive_Regulation : Positive_Regulation : Positive_Regulation
Positive_Regulation : Negative_Regulation : Positive_Regulation : Localization
```

Listing 5.4: Simplified form of the raw-signature in Listing 5.3. All non-event type terms are removed. In this case it is only the *THEME* and the *CAUSE* keyword.

2. Generalizing of Event Types
On top of this reduction, we assume that similar event types can be generalized without losing too much information. A modified GENIA event-scheme that we use in our approach, is displayed in Table 5.12. It displays all reductions and generalizations. The generalization comprises the encapsulation of *Gene expression* and *Transcription* into *Expression* and all forms of regulations to *Regulation.*

Generalized Event Types	Event Type	Core arguments	~~Additional arguments~~
Expression	Gene expression	~~Theme~~(Protein)	
	Transcription	~~Theme~~(Protein)	
Catabolism	Protein catabolism	~~Theme~~(Protein)	
Localization	Localization	~~Theme~~(Protein)	~~AtLoc(Entity)~~, ~~ToLoc(Entity)~~
Binding	Binding	~~Theme~~(Protein) +	~~Site(Entity) +~~
Regulation	Phosphorylation	~~Theme~~(Protein)	~~Site(Entity)~~
	Regulation	~~Theme~~(Protein / Event), ~~Cause~~(Protein / Event)	~~Site(Entity),~~ ~~CSite(Entity)~~
	Positive regulation	~~Theme~~(Protein / Event), ~~Cause~~(Protein / Event)	~~Site(Entity),~~ ~~CSite(Entity)~~
	Negative regulation	~~Theme~~(Protein / Event), ~~Cause~~(Protein / Event)	~~Site(Entity),~~ ~~CSite(Entity)~~

Table 5.12: Reduced GENIA event scheme.

The simplification comprises the deletion of all additional arguments and tree depending identifiers like *Theme* and *Cause*. The additional generalization transforms the signatures into the following form:

```
Regulation:Regulation:Regulation:Regulation
Regulation:Regulation:Regulation:Localization
```

Listing 5.5: Simplified and generalized form of the raw-signature from the example in Listing 5.3.

Signature String Compression The third and last step of the signature generation is a compression of the current form. On top of these simplifications and generalizations, which are based on assumptions, we use one last compression strategy to shorten the string length of very complex signatures. This strategy does not change the content of information, but is just a different spelling and leads to a condensation of information. We collapse consecutive equal relation-types and assign them a new term. These terms are composed of the prefix *Multiple*, the actual generalized event-type e.g. *Regulation* and an identifier that counts the number of collapsed events e.g. *_2*. Listing 5.6 shows the final forms of the modified signatures from Listing 5.3.

```
MultipleRegulation_4
MultipleRegulation_3:MultipleLocalization_1
```

Listing 5.6: Final form of the two Path Signatures from the example in Listing 5.3.

After this compression, the generation of a Path Signature is complete. In this form they will be used to define the features in Section 5.5.2.

5.4 The Graph Database

In the last two sections, the data extraction and preparation has been described. That includes the entity recognition, the entity normalization, the data filtering, and the data generation to build a comprehensive gene-interaction network. The extracted data need to be saved persistently in order to guarantee an access at query time. In this section, we describe the population of the databases. Thereto, the choice of the database is motivated in the following subsection. After that, the required data structure is explained and we describe the conversion of the previously prepared data into this format.

5.4.1 Blazegraph

This section describes and motivates briefly the choice of the database that was used to store the extracted information persistently. The choice of a (Resource Description Framework [13]) RDF-graph database is based on the fact that both, gene-disease and gene-gene relations can be represented as a triple. The database needs to connect all genes through their relations, not only with direct relations, but among longer paths. Therefore, the database needs to support graph mining techniques. To address all these requirements, we decided to use *bigdata* (also known as *Blazegraph* since 2015), which is maintained by Systap [7]. Blazegraph is a freely available database framework. It is highly scalable up to 50 billion entries on a single machine, which makes it easily usable, even with limited resources. *Blazegraph* stores each entry as resource description RDF-triple but supports also named graph quadruples. Named graphs offer also the possibility to add additional context knowledge in future work without changing to much of the data structure. The data can be accessed by using powerful low level API's like Sesame [14]. Sesame supports several query languages. This includes SPARQL, which was accepted as a standard by the World Wide Web Consortium (W3C). Thus, a lot of well documented examples exist that simplify the development of our system.

5.4.2 Population of the Database

To store the extracted information, the data needs to be converted into RDF-triples/ RDF-quadruples. This process is described in this section. An RDF triple contains three fields, namely *Subject*, *Predicate* and *Object*. A named graph quadruple contains an additional field, the *Context*. Such a triple describes a relation between the *Subject* and the *Object*. The *Predicate* covers their relation. Each field can store different data types and usually has a semantic meaning and provides predefined values. However, there is no restriction to use the fields in a different way.
In our case, we need to store three different types of triple.

1. The first type of triple stores an abstract based co-occurrence of a disease and a gene. We call this type of triple *co-occurrence triple* (COT).
2. The second type of triple stores the event-based relation between two genes. We name it *gene-event triple* (GET).

[7]`http://www.blazegraph.com/` Access: 23.04.15

3. The last type of triple is similar to the first type. It stores a relation between a disease and a gene. However, this relation is not based on co-occurrences, but is determined through the gold data. We name them *gold-data triple* (GDT).

Construction of the Co-occurrence Triples Each COT contains a disease, a gene and the PubMed-ID in which both entities were extracted. Given a set of diseases D and a set of genes G, which are extracted from the same abstract A, we built a set of triples $\{d, A, g | d \in D, g \in G\}$ to determine all COTs for A. Thus, the total number of COTs of A can be calculated as $|D| \cdot |G|$. This abstract based co-occurrence is thereby a very broad definition. However, as motivated in the related work chapter, most of the published findings describe successful experiments. Thus, we assume that the majority of abstract based co-occurrences share a positive correlation. By taking all co-occurrences into account, we avoid the missing of possible relations and increase thereby the systems recall. On the other hand, we create a lot of false positives that need to be eliminated. Table 5.13 contains an excerpt of all co-occurrences that were found in the example sentence in Example 5.8.

Disease (*Subject*)	AbstractID (*Predicate*)	Gene (*Object*)	ContextID (*Context*)
breast cancer	23848600	Act D	C125577
breast cancer	23848600	caspase 3	C125578
breast cancer	23848600	caspase 7	C125579
breast cancer	23848600	HSP27	C125580
sarcomas	23848600	Act D	C125581
sarcomas	23848600	caspase 3	C125582
⋮	⋮	⋮	⋮
sensitizes rhabdosarcoma	23848600	HSP27	C125588

Table 5.13: Excerpt from all COTs that can be extracted from the given sentence in Example 5.8.

We built a COT as follows: The disease is put into the *Subject* field. The *Predicate* field is filled with the abstract ID, since its meaning comes closest to a relation. Thereby, the *Object* contains the related gene. In the last field we store an entry corresponding context ID. This ID serves as pointer to the current entry. In future work we will be able to use this ID as a reference to store additional information to the corresponding COT. This could be for example the onset/offset of the involved entities or information about their surface forms to make the normalization process re-traceable.

Construction of the Gene Event Triples Each GET consists of a cause-gene, a theme-gene and their event-based relation. The main process of event generation was already described in Section 5.2. The generated triples for each extracted sub-tree as illustrated in Figure 5.5 and 5.6, are shown in Table 5.14.

Cause (*Subject*)	Complex Relation (*Predicate*)	Theme (*Object*)	ContextID (*Context*)
HSP27	THEME:Neg_reg:CAUSE:Pos_reg:THEME:Pos_reg:CAUSE	Act D	E5311
HSP27	THEME:Neg_reg:CAUSE:Pos_regTHEME:Pos_reg:THEME	caspase 3	E5315
Act D	CAUSE:Pos_reg:THEME	caspase 3	E5316

Table 5.14: This table shows all generated triples that could be generated from the sentence in Example 5.8. Due to lack of space we shortened the event type names. Thereto, *Neg_rel* stands for *Negative_regulation*, *Pos_rel* can be analogously interpreted.

The amount of gene relations in this example is substantially lower in comparison to the number of co-occurrence triples. There are only three different GETs. The data allocation to the individual fields is similar to the co-occurrence triple. The cause-gene is assigned to the *Subject* field. The relation is stored as *Predicate*. The theme-gene serves as *Object*. The corresponding context ID is stored in the *Context* field and can be interpreted analogously to the co-occurrence context ID.

Construction of the Gold Data Triple The third type of triple covers an entry in the gold data that were previously described in Section 3.3. The structure and meaning of a gold triple is very similar to a co-occurrence triple. It contains exactly one disease and one gene. They only differ in the predicate. A COT stores the abstract ID in which both entities were found and the GDT contains the information from which particular gold data this association was extracted. In the current system, this could be either *ncbigtr* or *disgenet*. Table 5.15 shows six example GDTs.

Technically, all three types of named triples can be stored in a single comprehensive database. However, we decided not to do so, but to store the event type triples in its own database. The co-occurrence triples were put together with the gold triples. This split has benefits and drawbacks. During the design of the feature vector, a clear distinction into features that are based on COTs /GDTs and features that are based on GETs emerged. Thus, a split into two databases seems legit. It will ease the way of querying data and speed up the process. A clear negative aspect is the higher complexity of querying any kind of interaction between COTS and GETs. This requires both, more time and resources since pre-queried data need to be stored temporarily in the memory.

Disease (*Subject*)	Gold Data (*Predicate*)	Gene (*Object*)	ContextID (*Context*)
hyperlysinemia	ncbigtr	AASS	GTR13
malignant hyperthermia	ncbigtr	RYR1	GTR4605
townes syndrome	ncbigtr	SALL1	GTR4616
malignant hyperthermia	disgenet	RYR1	Dis52
pulmonary fibrosis	disgenet	SFTPA2	Dis2141
endometriosis	disgenet	MMP3	Dis42577

Table 5.15: Six example triples that describe gold entries from the GTR respectively from the DisGeNET gold data.

Another problem could appear in the future. A visualization of the entire data is not simple. Usually there are a lot of visualization tools for RDF-data like *vizualRDF*[8], *RDF-Gravity*[9] or *IsaViz*[10]. However, a split into two databases makes a comprehensive visualization more complicated.
Finally, the positive aspects predominate and we decided to populate two separate databases. Thus, all COTs and GETs are stored in the *disease-genes-association*-database, hereinafter referred to as DGA-DB. All event triples are stored in the *gene-interaction-network*-database, referred to as GIN-DB.

Insertion of the triples In the following we give an example query to store a co-occurrence quadruple.

```
PREFIX c: cooccurrence

INSERT DATA
  {
    GRAPH <c:CO> { <c:breast+cancer> <c:23848600> <c:Act+D> }
  }
```

Listing 5.7: Concrete co-occurrence quadruple insertion into the DGA-DB for the disease *breast cancer* and the gene *Act D* in SPARQL-notation.

[8]`https://github.com/alangrafu/visualRDF` Access: 2015-04-15
[9]`http://semweb.salzburgresearch.at/apps/rdf-gravity` Access: 2015-04-15
[10]`http://www.w3.org/2001/11/IsaViz` Access: 2015-04-15

The *Resource Description Framework* was technically invented to describe semantic web data. Usually, such data is provided with a namespace that declares the scope in which the attribute value is defined. In the current system, we do not make use of the benefits that can be gained by choosing accurate namespaces. Nonetheless, the data is stored in URI-notation and a namespace declaration is mandatory. The example in 5.7 shows an insertion for a co-occurrence triple into the DGA-DB. The triple for the gold-data and the gene-events can be inserted analogously. They differ only in the namespace *gold* for gold-data triples and *event* for gene-event triples.

5.5 Feature Extraction

In the last sections, the data generation and the persistent storage of the extracted information was described. The data could be separated into three different kinds of named RDF-triple. We populate a separate database for the gene-gene associations and a separate for the disease-gene associations. This was motivated by the investigation of the formalized features. As we will see in this section, the triple separation improves the simplicity of querying the data. We formalize all features and describe the methods to extract them from the two databases. Based on these features, we build a comprehensive feature vector that represents a data point to train machine learning models. Finally, the main goal of this work, the gene classification and the gene ranking can be tackled.
Seven feature groups are defined in total. Each feature group adds a number of features to the comprehensive feature vector that is used as training data for the machine learning methods. This arrangement in groups can again be divided into two types.

The first type is based on disease-gene pairs. These features are based on the co-occurrence between diseases and genes. We call these feature groups *co-occurrence based features*. They include the following five feature groups:

1. *Entropy* - The Entropy feature group contains the entropy of the gene and the entropy of the disease.
2. *Co-occurrence* -It contains three features that describe the probability of the disease-gene pair with respect to three different types of normalizations.
3. *Grade* -The grade describes the normalized occurrence of a gene or a disease in the database.
4. *Odds Ratio* -The odds ratio describes the statistical dependency between a gene and a disease within the dataset.

5. *TF-IDF* - This feature contains the TF-IDF value for a disease-gene pair. A Term is defined by a gene that co-occurs with a disease (the document).

The second type of feature groups is called *gene network features.* These features are disease independent and based only on the gene-interaction network. This type contains the following feature groups:

1. *Path Signatures* - A signature describes a path between a given cause-gene and a theme-gene in the gene-interaction network as defined in Section 5.3.2. We consider each outgoing signature with a specific length as single feature. The actual value is defined by the TF-IDF value of a signature within the corresponding gene.
2. *Gene Connectivity* - These features contain information about the connectivity of the gene in the network with respect to the number of outgoing and incoming signatures.

5.5.1 Co-occurrences Based Features

In the following subsection we describe all features that are based on co-occurrences between diseases and genes.

Before formalizing these features, some variables need to be predefined. We denote $T_{diseases}$ as the set of all triples within DGA-DB. Further we define D as the set of all existing diseases and G as the set of all existing genes in DGA-DB.

On top of that we define three subsets of $T_{diseases}$:

1. T_d = Subset of triples with a fix disease d and all co-occurring genes.
2. T_g = Subset of triples with a fix gene g and all co-occurring diseases.
3. T_{dg} = Subset of triples with a fix disease d and a fix gene g.

where $T_{dg} \subseteq T_d$, $T_{dg} \subseteq T_g$ and $T_{dg} = T_d \cap T_g$ as illustrated in Figure 5.10.

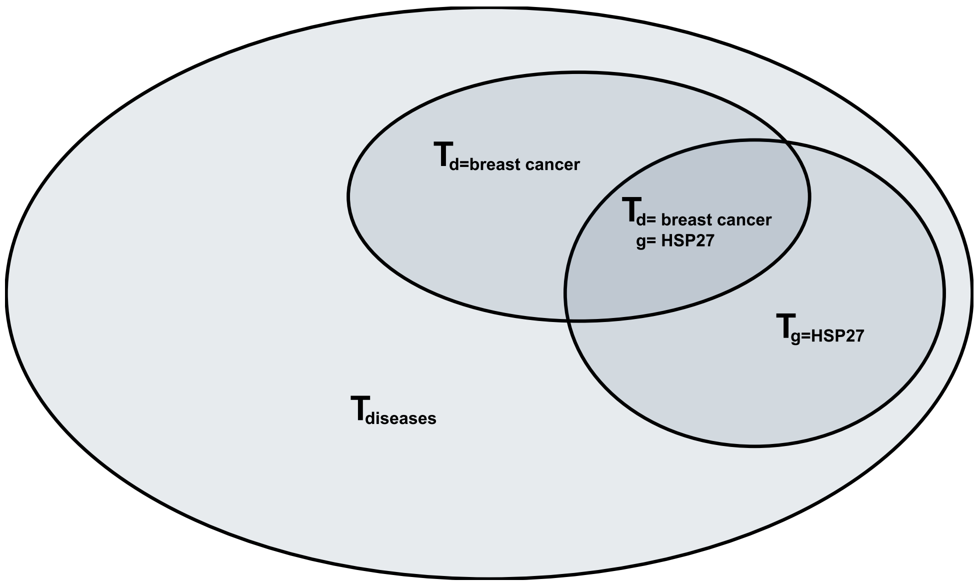

Figure 5.10: Illustration of the three defined subsets: T_d,T_g and T_{dg} of $T_{diseases}$ for the example disease *breast cancer* and the gene *HSP27*.

To calculate these subsets, we used three SPARQL queries that are shown in Listing 5.10,5.9 and 5.10. The first query extracts all co-occurring genes for the example disease *breast cancer* and is denoted as $T_{d=breastcancer}$:

```
PREFIX c: cooccurrence
PREFIX g: gold

SELECT DISTINCT ?Rel, ?Gene WHERE
  {
    <c:breast+cancer> ?Rel ?Gene .
    FILTER(?Rel != <g:disgenet> && ?Rel != <g:ncbigtr> )
  }
```

Listing 5.8: SPARQL query to retrieve a list of co-occurring genes for the specific disease *breast cancer* in URL notation. The result is filtered to receive entries that do not belong to any gold standard.

In this query we add a filter in line seven to restrict the output to non-gold standard data. Since we have two gold standards as described in Section 3.3, we need to filter both. As described in Section 5.4, the source of a disease-gene association can be identified by their predicate. Thus we filter all entries whose relation is either *disgenet* or *ncbigtr*. The result is a list of relation-gene tuples that co-occur with *breast cancer*.

The extraction of all co-occurring diseases for the gene *HSP27* ($T_{g=HSP27}$) works analogously.

```
PREFIX c: cooccurrence
PREFIX g: gold

SELECT DISTINCT ?Dis ?Rel WHERE
  {
    ?Dis ?Rel <c:HSP27> .
    FILTER(?Rel != <g:disgenet> && ?Rel != <g:ncbigtr> )
  }
```

Listing 5.9: SPARQL query to retrieve a list of co-occurring diseases for the specific gene *HSP27*. The result is filtered to receive entries that do not belong to any gold standard.

The last query that is used extracts all co-occurrences for *breast cancer* and *HSP27*. Since we already specified the disease and the gene, the result is simply a list of relations, which is denoted as $T_{d=breastcancer,\ g=HSP27}$.

```
PREFIX c: cooccurrence
PREFIX g: gold

SELECT DISTINCT ?Rel WHERE
  {
    <c:breast+cancer> ?Rel <c:HSP27> .
    FILTER(?Rel != <g:disgenet> && ?Rel != <g:ncbigtr> )
  }
```

Listing 5.10: SPARQL query to retrieve a list of relations that connect the specific diseases *cancer* with the specific gene *LecB*. The result is filtered to receive entries that do not belong to any gold standard.

In the following, each feature group is described in detail by using the predefined sets. We denote $d \in D$ as the disease and $g \in G$ as the gene of a given disease-gene pair to which the features should be calculated.

Entropy

In an event that is determined by the co-occurrence of two variables, the entropy measures the amount of uncertainty about the outcome of the event that remains when one variable is known. Following the definition of Shannon's Entropy [57], we establish two formulas that calculate the entropies for d, respectively g.

In the following equation, the calculation of the entropy $H(d)$ for d is shown:

$$H(d) = -\sum_{g}^{G} p_g(g,d) \cdot \log_2(p_g(g,d)) \tag{5.11}$$

We define $p_g(g,d)$ as the probability at which g co-occurs with d within the data:

$$p_g(g,d) = \frac{|T_{dg}|}{\sum_{g'}^{G} |T_{dg'}|}$$

The calculation of the entropy $H(g)$ for the gene g works analogously as shown in Equation (5.12).

$$\begin{aligned} H(g) &= -\sum_{d}^{D} p_d(d,g) \cdot \log_2(p_d(d,g)) \\ p_d(d,g) &= \frac{|T_{dg}|}{\sum_{d'}^{D} |T_{d'g}|} \end{aligned} \tag{5.12}$$

Two real world examples for *INS* and *LecB* are shown in Table 5.16. The first column counts the number of diseases that co-occur with the gene. The second column shows the corresponding entropy values. *INS* is a very frequent gene that co-occurs with more than 8.000 different diseases. The entropy shows a relative high value of 10.47. In contrast, *LecB* co-occurs with only 7 different diseases. Its entropy value of 2.73 is rather small. These values are consistent with the definition of Shannon's entropy. A gene that co-occurs with many different diseases has a high entropy and thereby leads to a high uncertainty. More specific genes have small entropies and thereby low uncertainties to which disease it could co-occur. This feature group is implemented to offer the possibility of balancing the importance between rare and frequent genes, respectively diseases.

	Mentioned Diseases	Entropy
INS	8.853	10.47
LecB	7	2.73

Table 5.16: Example of the entropy for two different genes.

Co-occurrence

The co-occurrence feature group describes the probability of a specific disease-gene pair with respect to three different normalizations. The first feature value $Occ_{dg}(d, g)$ is calculated by the number of co-occurrences between d and g and is normalized by the maximum disease-gene co-occurrence that exists in the data:

$$Occ_{dg}(d, g) = \frac{|T_{dg}|}{max\{|T_{d'}| \ : \ d' \in D\}} \tag{5.13}$$

The second feature $Occ_{d}(d, g)$ value is normalized by the number of co-occurrences that exist between the fixed disease d and any gene $g' \in G$ where $g' \neq g$.

$$Occ_{d}(d, g) = \frac{|T_{dg}|}{|T_{d}| - |T_{dg}|} \tag{5.14}$$

Analog to the second feature, the third one is normalized with respect to a fixed gene and any disease $d' \in D$ where $d' \neq d$.

$$Occ_{g}(d, g) = \frac{|T_{dg}|}{|T_{g}| - |T_{dg}|} \tag{5.15}$$

These features enable several points of view to the information that can be gained by the specified disease-gene pair. The first feature $Occ_{dg}(d, g)$ denotes the general strength of their connection. The values range from zero to one. A value near to zero indicates a weak connection, whereas a value near to one indicates a strong connection. The second and third features define, similar to the entropy, the individual specificity of the co-occurrence with respect to the disease respectively to the gene. The higher the value, the more specific is the particular disease connected to the gene respectively the gene connected to the disease.

Grades

The *Grades* feature group consists of two single features. The first feature counts the number of triples that contain d. The second feature counts the number of triples that contain g. Both values are normalized by their corresponding maximum value. Equation (5.16) and (5.17) show the formulas to calculate both grade features.

$$Grade_d(d) = \frac{|T_d|}{max\{|T_{d'} : d' \in D\}} \tag{5.16}$$

$$Grade_g(g) = \frac{|T'_g|}{max\{|T'_{g'} : g' \in G'\}} \tag{5.17}$$

The grade feature is another way to express the frequency of an entity. The higher the grade, the more frequent is the specific entity.

Odds Ratio

The odds ratio describes a statistical way to measure the dependency between two attributes within a dataset. We use this feature to describe the dependency between d and g. The first attribute A is defined as the presence or absence of d within a triple. The second attribute B works analogously with respect to g. By observing these attributes, a matrix can be set up with four elements as shown in Table 5.17. Each field in the matrix counts the number of pairwise occurrence of the attributes.

	B	¬B
A	$\|T_{dg}\|$	$\|T_{dg}\| - \|T_g\|$
¬A	$\|T_{dg}\| - \|T_d\|$	$\|T_{diseases}\| - \|T_{dg}\|$

Table 5.17: Odds ratio attribute matrix. With the two attributes A and B corresponding to the presence or absence to the disease (A) respectively the presence or absence of the gene (B).

The first field AB counts the number of triples in which the disease and the gene co-occur. $A\neg B$ represents the number of co-occurrences where d co-occurs with any gene except g. $\neg AB$ can be interpreted analogously. The last field $\neg A\neg B$ contains the number of all co-occurrences that neither contain the disease nor the gene.

To calculate the odds ratio, we use the formula defined in Equation (5.18).

$$Odds_{gd}(d, g) = \frac{(AB) * (\neg A\neg B)}{(A\neg B) * (\neg AB)} = \frac{|T_{dg}| * (|T| - |T_{dg}|)}{(|T_{dg}| - |T_d|) * (|T_{dg}| - |T_g|)} \tag{5.18}$$

In theory, the result is a number between zero and ∞. However, since we have much more triples that match $\neg A\neg B$ than the other three conditions, the odds ratio is either greater than one or exactly zero. Thereby we interpret the value as follows: The higher the odds ratio the higher, the dependency between d and g. A value of zero can only be reached if the *(AB-term)* is zero, which means there is no dependency.

TF - IDF

The term frequency (TF) and the inverse document frequency (IDF) are measures that are often used in the task of information retrieval from texts. From a given set of textual documents, the TF-IDF is the information that can be gained from a word in a document in order to classify the document. We adapted this way of information retrieval to the problem of weighting a disease-gene association. By that, we do not define a document as textual source, but as a disease that contains co-occurring genes. Thus, a term within this document is defined as one co-occurring gene. By investigating the considered data, we noticed a very high variance in the number of genes that are co-occurring with a disease. Especially rare occurring diseases often only have a few co-occurring genes, whereas frequently occurring diseases like cancer are mentioned with up to several hundred genes. To avoid a bias towards longer documents (diseases that co-occur with many genes), the term frequency is calculated by using the augmented normalization, which is calculated as shown in Equation (5.19).

$$tf_{aug}(g,d) = 0.5 + \left(0.5 * \frac{|T_{dg}|}{max\{|T_{dg'}| : g' \in G\}}\right) \tag{5.19}$$

The inverse document frequency is defined in Equation (5.20).

$$\begin{aligned} idf(g,D) &= log(\frac{|D|}{\sum_d^D f(d,g)}) \\ f(d,g) &= \begin{cases} 1 & if\ |T_{dg}| > 0 \\ 0 & else \end{cases} \end{aligned} \tag{5.20}$$

The final feature value $tfidf$ is then calculated as the product of the single components:

$$tfidf(g,d,D) = tf_{aug}(g,d) * idf(g,D) \tag{5.21}$$

The final contribution to the feature vector are two features, namely the TF-IDF value but also the IDF value separately.

5.5.2 Gene Interaction Based Features

All previously described feature groups take a disease and a gene into account. The next two feature groups, the *Path Signatures* and the *Gene Connectivity* are disease independent and purely calculated on the gene-interaction network. Thereto, we define T_{genes} as a set of all triples in GID-DB.

Path Signatures

In this group, each feature represents a unique outgoing signature, as described in Section 5.3.2. Thereby, the number of possible features depends on both, the number of different genes and the maximum path length respectively the maximum number of bridge-genes. Since the number of possible features grows exponentially by increasing one of these two factors, we set the maximum path length to two, respectively the maximum number of bridge-genes to one. The limitation of the number of genes cannot be restricted that easily. In fact, there are only two ways to reduce the number of genes effectively. The first and most important reduction is done by an effective gene-normalization framework. An other way is to reduce the genes by limiting the taxonomic scope. This could be for example a limitation to human genes as we did it in this work. On the one hand, each limitation could lead to missing a significant structure in the data, which could easily have a negative effect on the final results. On the other hand, some limitations and assumptions need to be made in order to reduce the processing time or to focus the results to the desired scope. A well defined purpose of the data usage is thereby crucial and should be done very carefully in the first place.

The signature-feature value depends on two variables, a gene g and a signature s. The actual value is defined by the TF-IDF value of the term s within the document g as shown in the equations (5.22), (5.23) and (5.24). We define $S_{out}(g)$ as a list of all outgoing signatures (as described in Section 5.3.2) for the specified gene g. Notice that, due to the fact that the signature are gene-independent but not their generation, these lists could contain any signatures multiple times.

The basic idea of these features is to find one or more specific signatures, that may identify an important relation between a gene that is already known as disease-related and a new gene. Currently, only the outgoing signatures are considered in this feature group, since we assume that outgoing signatures contribute more relevant information. In a further improvement of the system, we should extend these features by incoming signatures as well.

$$tfidf_{sig}(s,g,G) = tf_{sig}(s,g) * idf_{sig}(s,G) \tag{5.22}$$

$$\begin{gathered} tf_{sig}(s,g) = \frac{\sum_{s'}^{S_g^{out}} f(s,s')}{max\{\sum_{s'}^{S_{out}(g)} f(s,s') : s \in S_{out}(g)\}} \\ f(s,s') = \begin{cases} 1 & if\ s = s' \\ 0 & else \end{cases} \end{gathered} \tag{5.23}$$

$$\begin{gathered} idf_{sig}(s,G) = log(\frac{|G|}{\sum_g^G f(g,s)}) \\ f(g',s) = \begin{cases} 1 & if\ s \in S_{out}(g') \\ 0 & else \end{cases} \end{gathered} \tag{5.24}$$

Gene Connectivity

The last feature group describes the connectivity of a gene within the gene-network graph. This group contains four different types of features. The number of single features depends on the maximum number of bridge-genes. The first features count the number of outgoing signatures for each length separately. The next features count the number of incoming signatures for each length separately. The last two features, represent the arithmetic mean, respectively the ratio of the previous features.

To calculate these features we denote $S_{in}(g,l)$ as a list of all incoming signatures and $S_{out}(g,l)$ as a list of all outgoing signatures with the specific path length of l. Further, we denote L as the maximum path length where $l \leq L$. The query to extract all outgoing signatures for a specific gene e.g. *HSP27* is similar to that one defined in Listing 5.2. The only difference is that we do not specify a theme-gene. The query to create $S_{out}(g,l=2)$

for *HSP27* is shown in Listing 5.11.

```
PREFIX e: event
SELECT ?OutRel1, ?OutRel2  WHERE
  {
    <e:HSP27> ?OutRel1 ?BridgeGene1 .
    ?BridgeGene1 ?OutRel2 ?GeneTheme1 .
  }
```

Listing 5.11: SPARQL query to retrieve a list of all outgoing relations for the cause-gene *HSP27*.

These extracted relations can be used to generate an outgoing signature by following the description in Section 5.3.2. The generation of $S_{in}(g, l = 1)$ is analogously. Listing 5.12 shows an example query to retrieve all incoming signatures for the specific theme-gene *HSP27*.

```
PREFIX e: event
SELECT ?InRel  WHERE
  {
    ?CauseGene ?InRel <e:HSP27> .
  }
```

Listing 5.12: SPARQL query to retrieve a list of all incoming relations for the theme-gene *HSP27* with a length of 1.

The first feature $Out_{norm}(g, l)$ and the second feature $In_{norm}(g, l)$ can be calculated as shown in the following equations:

$$Out_{norm}(g, l) = \frac{|S_{out}(g, l)|}{max\{|S_{out}(g', l)| : g' \in G\}} \tag{5.25}$$

$$In_{norm}(g) = \frac{|S_{in}(g, l)|}{max\{|S_{in}(g', l)| : g' \in G\}} \tag{5.26}$$

The third feature, $Presence(g)$ represents the normalized mean value and is calculated by:

$$\begin{aligned} Presence(g) &= \frac{Out(g) + In(g)}{max\{(Out(g') + In(g'')) : g', g'' \in G\}} \\ Out(g) &= \sum_{l}^{L} |S_{out}(g,l)| \\ In(g) &= \sum_{l}^{L} |S_{in}(g,l)| \end{aligned} \tag{5.27}$$

In all three features, the values range from zero to one. The higher the value, the higher the connectivity of g in the network.

The fourth feature $IORatio(g)$ is defined by the ratio between the incoming and outgoing number of signatures. The ratio is defined as:

$$IORatio(g) = \frac{max(1, Out(g))}{max(1, In(g))} \tag{5.28}$$

If the result is greater than one, the gene has more outgoing edges and has a rather manipulating role in the gene network. In contrary, if the number is less than one, it indicates that the gene is more manipulated by other genes. If the specific gene does not have either an outgoing or an incoming signature, we set the corresponding value to one. Thus, a division by zero is avoided. With a maximum path length of two, this feature group has six features that contribute information about the gene-connectivity.

5.5.3 The Feature Vector

In the previous subsections, we formalized all features and explained their generation and calculation in detail. In this subsection, the composition of the feature vector is described.

1. *Entropy* : $SV_{entropy}$ - The sub vector for the *Entropy* is a two dimensional vector:

$$SV_{entropy}(d,g) = \begin{pmatrix} H(d) \\ H(g) \end{pmatrix} \tag{5.29}$$

   The values are calculated by Equation (5.11) and (5.12).

2. *Co-occurrence : SV_{cooc}* - The sub vector for the *Co-occurrence* is a three dimensional vector:

$$SV_{cooc}(d,g) = \begin{pmatrix} Occ_{dg}(d,g) \\ Occ_{d}(d) \\ Occ_{g}(g) \end{pmatrix} \tag{5.30}$$

   The values are calculated by Equation (5.13), (5.14) and (5.15).

3. *Grade : SV_{grade}* - The sub vector for the *Grade* is a two dimensional vector:

$$SV_{grade}(d,g) = \begin{pmatrix} Grade_{d}(d) \\ Grade_{g}(g) \end{pmatrix} \tag{5.31}$$

   The values are calculated by Equation (5.16) and (5.17).

4. *Odds Ratio : SV_{odds}* - The sub vector for the *Odds Ratio* is a one dimensional vector:

$$SV_{odds}(d,g) = \begin{pmatrix} Odds_{gd}(d,g) \end{pmatrix} \tag{5.32}$$

   The value is calculated by Equation (5.18).

5. *TF-IDF : SV_{tfidf}* - The sub vector for the *TF-IDF* is a two dimensional vector:

$$SV_{tfidf}(d,g) = \begin{pmatrix} tfidf(g,d,D) \\ idf(g,D) \end{pmatrix} \tag{5.33}$$

   The values are calculated by Equation (5.20) and (5.21).

6. *Gene Connectivity : $SV_{connectivity}$* - The *Gene Connectivity* sub vector with a maximum path length of two is a six dimensional vector:

$$SV_{\text{connectivity}}(g) = \begin{pmatrix} Out_{norm}(g,1) \\ Out_{norm}(g,2) \\ In_{norm}(g,1) \\ In_{norm}(g,2) \\ Presence(g) \\ IORatio(g) \end{pmatrix} \tag{5.34}$$

The values are calculated by Equation (5.25), (5.26), (5.27) and (5.28).

7. *Path Signatures : $SV_{signatures}$* - The dimension of the sub vector for the *Path Signatures* depends on the considered genes and the maximum path length. The latter variable needs to be defined in the first place and must not be changed anymore. The maximum path length was set to two. The genes were restricted to those, who occur in the currently used gold standard. The features for the *Path Signatures* group were then aggregated for each of those genes. Given G_{gold} defined as a set of all genes that occur in the gold standard, the total amount of different signatures/features denoted as F_{sig} is calculated as:

$$F_{sig} = \{S_{out}(g,l) : g \in G_{gold}, l \leq L\} \tag{5.35}$$

Thereby, the sub vector for the *Signatures* is a n dimensional vector, where $n = |F_{sig}|$:

$$SV_{signatures}(g) = \begin{pmatrix} tfidf_{sig}(F_{sig}[0], g) \\ tfidf_{sig}(F_{sig}[1], g) \\ tfidf_{sig}(F_{sig}[2], g) \\ \vdots \\ tfidf_{sig}(F_{sig}[n-1], g) \end{pmatrix} \tag{5.36}$$

The values are calculated by Equation (5.22).

The final feature vector FV for a given disease d and a given gene g is then a composition of all sub vectors:

$$FV(d,g) = \begin{pmatrix} SV_{entropy}(d,g) \\ SV_{cooc}(d,g) \\ SV_{grade}(d,g) \\ SV_{odds}(d,g) \\ SV_{tfidf}(d,g) \\ SV_{connectivity}(g) \\ SV_{signatures}(g) \end{pmatrix} \tag{5.37}$$

with a total of $14 + |F_{sig}|$ dimensions.

5.6 Summary

In this chapter, we discussed the most important methods that we use in our approach. In Section 5.1 we described the disease extraction pipeline. This pipeline consists of two consecutive methods, the disease mention recognition and the disease normalization. We integrated a CRF tagger by Klinger [36] as disease recognition. In Section 5.1.2, we introduced DiNo, a simple disease normalization framework. We evaluated the entire pipeline on the NCBI disease corpus [23] and archived an F-score of 61.79%.

The gene event extraction pipeline was discussed in Section 5.2. The pipeline starts with the event extraction system TEES [9]. All genes that are found by TEES are normalized by GeNo [72]. We evaluated the combination of both state-of-the-art systems on the BioCreative-II gene normalization corpus [46]. The usage of TEES as gene recognition model for GeNo yields a performance of 55.33%. During the normalization, we discarded all genes that did not belong to the human species. We use EntrezGene [45] to accomplish this filtering by comparing the taxonomic IDs.

By that, we heavily reduced the amount of considered genes and events. On top of that, we transformed the remaining complex gene-events into simpler sub-events, describing a relation between two genes. After an extensive graph transformation, the relations were then simplified and normalized in order to reduce their complexity. The entire process of event graph generation was described in Section 5.3. For the task of feature generation, we defined a *Path Signature* as modified consecutive relations between two genes in the gene-interaction network. The process of signature generation was described in Section 5.3.2.

Based on the gold-standard data, which was introduced in Section 3.3, and the extracted diseases, genes and gene-events, we store three different types of RDF-quadruple in two different databases. The first kind of triple covers an abstract-based co-occurrence of a disease and a gene. The next type of triple contains a disease-gene association from the gold data. Both types of triples were stored together in the *Disease-Gene-Association* database. The third type of triple covers a simplified gene-event. By integrating all extracted gene-events into a graph database, we generated a comprehensive gene-interaction network where genes are nodes and their relations are expressed through edges. We store the third type of triple in a separate database called *Gene-Interaction-Network* database. The triple generation and storage were described in Section 5.4.

In Section 5.5, we formalized the features that we used to train machine learning models. We distinguished between features that are based on the *Disease-Gene-Association* database and those that are based on the *Gene-Interaction-Network* database. The first kind included the following five features: *Entropy*, *Co-occurrence*, *Grade*, *Odds Ratio* and *TFIDF*. These features take a specific disease and a specific gene into account. The second kind of features are disease independent and are calculated on the gene-interaction network. These are: *Path Signatures* and *Gene Connectivity*. The chapter ends with a comprehensive view on the final feature vector that can be generated for any disease-gene pair using the presented methods in this chapter.

6 Experiments and Evaluation

This work describes a system for disease-gene association extraction, with respect to the identification of undiscovered public knowledge. We addressed this task by building and analyzing a gene-interaction network. The aggregation of information of gene-interactions that goes beyond the scope of a single abstract, allows us to generate novel disease associated gene candidates. Using concept-interaction networks to find hidden knowledge, often leads to a high number of novel candidates. In order to deal with this problem, we implement two different methods. In the first method, we train a support vector machine to classify a gene-candidate as disease related or not. In our second method, we train a support vector regression to rank gene-candidates by their importance to a disease. Both methods have advantages and disadvantages, which will be described in the following experiments. However, both methods offer the possibility to limit the amount of novel candidates and focus the attention to the most important.

So far we have discussed:

1. the materials that we used for the information extraction in Chapter 3.
2. the system architecture in Chapter 4 that gives an insight of the resource generation pipeline.
3. the most important methods of the resource generation in detail. And we formalized various features to define the feature space, in Chapter 5.

In this chapter, we present several experiments to test the performance of our system. In the first section (Section 6.1), we describe the experiments we have made to find the best classification model. We trained various models that were built with different features-sets and evaluate them on a test set. In Section 6.2, we describe experiments to find the best regression model. On top of that, we integrate the best performing regression model of the training data into our system, and pre-evaluate the ranking task on a test set. In a last experiment, presented in Section 6.3, we examine our system's performance with a case-study. We use *Pulmonary Fibrosis* as testing disease and generate 200 gene-candidates, which were evaluated by biomedical experts.

During the case-study, we want to prove the following assumption:

(6.1) *We assume that a gene-interaction network, such as we built in Section 5.3, contains reliable features that contribute useful knowledge, which can be learned by machine learning methods, to enhance the quality of the generated output.*

6.1 Classification of Genes Candidates

In this section, we describe the general experimental setup from our experiments, finding the best performing classification model. This includes a description of the training and test set, the measurement and a brief overview of the used machine learning method. In Section 6.1.2, we present and discuss the evaluation results. The main objective of these experiments was to find the best performing model that represents the *Genetic Testing Registry*-gold data, with respect of our assumption in Example 6.1.

6.1.1 Methods and Experimental Setup

Classification with Support Vector Machine In this work, we use a standard soft-margin Support Vector Machine [19] for classification (C-SVM), and use the well established *Gaussian*-kernel function, also known as *Radial Basis Function* (RBF), to deal with data that is not linear separable. The C-SVM is a common supervised machine learning method for binary classification. It classifies a data point $x \in \mathbb{R}^n$ by assigning a class label $y \in \{-1, 1\}$. Given a labeled training set of N data points $\{(x_i, y_i)|i \in N\}$, the goal of the C-SVM is to find a hyper-plane that separates binary-labeled data points with a minimum of false classifications and a maximum generalization. The efficiency of our C-SVMs depends on the choice of two parameters: the *gamma* parameter of the RBF and the soft-margin parameter C of the C-SVM. The actual implementation of an SVM was provided by LibSVM [15], which can be integrated into the WEKA-environment. For more information, we refer to [19, 63].

Meta Parametrization To optimize the classification results, we adjust the two parameters *gamma* and C:

- C is the parameter for the soft-margin cost function. It controls the influence of each individual support vector that defines the hyper-plane. A larger C aims at classifying all training examples correctly. By that, the SVM is able to select more training points as support vectors, which leads to a high variance due to penalizing miss-classifications a lot. A lower C allows the SVM to ignore miss-classified data points with respect to finding the maximum margin.

- *gamma* is the parameter of the Gaussian kernel function to handle non-linear classifications. It defines the influence of each training point to the hyper-plane. A low *gamma* leads to a far reach, a high *gamma* leads to a close reach.

C and *gamma* cannot be determined independently. Thus, to find the best trade-off between a large margin and a small error penalty, we perform a grid search within a fixed boundary for both parameters. We use the integrated grid-search function in WEKA [29], to select the best pair between 0.1 and 10000 (exploring in decimal powers).

Feature Reduction Besides the meta parametrization, we reduce the feature space to improve the classification and speed up the process. A common way to do this is to determine the *Information Gain* [37] (IG) for each feature, to rank them by their importance and chose only the best ones. The IG evaluates the worth of a feature with respect to its class as shown in the following equation, where H denotes Shannon's Entropy [57]:

$$\mathrm{IG}(class, feature) = H(class) - H(class|feature) \tag{6.2}$$

The IG can be interpreted as the reduction of the uncertainty of the prediction by taking a given feature into account. That means, the higher the value, the more worthy the feature. We used the integrated WEKA-function to calculate the IG, to get a list of features sorted by their worthy. We use the IG to extract the best n features, where n variates during the experiments, and thereby reduce the dimensionality of our feature space to n.

Macro Average F-Score The experimental results are measured and compared by determining the precision, recall and (harmonic mean) F-Score, which were calculated as follows:

$$\begin{aligned} p &= \frac{true\ positive}{true\ positive + false\ positive} \\ r &= \frac{true\ positive}{true\ positive + false\ negative} \\ f &= 2 \cdot \frac{p \cdot r}{p + r} \end{aligned} \tag{6.3}$$

We defined the number of true positives as all genes that were classified correctly as related to the specific disease. The false positives and false negatives are defined analogously. In these experiments, we consider the macro average results to evaluate the performance of the models. To determine the macro average results, the precision, recall and F-Score is calculated separately for each disease. Subsequently, the arithmetic mean is built over all diseases.

The Training and Test Set As mentioned before, we train a C-SVM to classify genes as disease related or not. Thereto, a training and a test set is created, using the *Genetic Testing Registry* as gold-standard data. All disease-gene associations in GTR are human curated. Thus, we assume that each association represents a positive data example. We labeled all associations from the gold-standard with the class value 1. For each disease in GTR, we then extract gene-associations from our previously built *disease-gene-association* database (see Section 5.4), hereinafter referred to as DGA-DB, to extract almost the same amount of negative training examples. A disease-gene-association within DGA-DB is a negative training example, if and only if the association is not mentioned in GTR. All negative examples are labeled with the class value -1. Once the positive and negative training examples are extracted, we divide the data into an 80-20 split. We use 80% as training data and 20% as test data. The training data consists of 3.665 data points with 1.781 negative and 1.884 positive examples. The test set consists of 910 data points with 440 negative examples and 470 positive examples. To ensure comparable results during the evaluation of the different models, we fixed these training and test sets.

6.1.2 Evaluation Results and Discussion

Our experiments on the classification models are separated into two parts. In the first part, we evaluate the models on the training data, using a ten-fold cross validation. The objective of this pre-evaluation is to extract the best and interesting models. These will be evaluated against the test set and will later be integrated into the entire pipeline. Each model contains a different compositions of features. They can be separated into *Feature Group*-models (FG), *Best N Signature Features*-models (BSF) and *Best M Features*-models (BMF). In Table 6.1, we compare all models in FG among themselves. Table 6.2 shows the results of all evaluated BSF models. Table 6.3 compares all models in BMF. Finally, the best models from each model group are compared with our baseline model on the test set. The results are shown in Table 6.4.

Feature Group Models A model in FG contains exact one feature group that was described in Section5.5. Beyond those single-feature-group models, FG includes a model that takes all co-occurrence based features into account (CBF) (as described in Section

5.5.1), and a model that takes all co-occurrences based features in combination with the gene-connectivity features (CBF+Connectivity) into account. We evaluate all models in FG, performing a ten-fold-cross validation on the training data. The results are presented in Table 6.1.

Feature Group	Precision	Recall	F-Score
Entropy	64.1	62.9	63.5
Co-occurrence	88.1	71.9	79.2
Grade	58.4	82	68.2
Odds Ratio	80.7	28.8	41.8
TF-IDF	91.3	65	75.9
Path Signatures	62.9	79.8	70.3
Connectivity	60.8	70.1	65.1
CBF	91.1	76.6	83.2
CBF+Connectivity	89.7	79.8	84.5

Table 6.1: Evaluation results of the FG-classification models on the training data using a ten-fold cross validation. The highest value for precision, recall and F-Score is highlighted.

The *Entropy* reaches an F-Score of 63.5% with a precision of 64.1% and a recall of 62.9%. The *Co-occurrence*-model is the best performing model that uses a single feature group, with an F-Score of 79.2%, which is a composition of the high precision of 88.1% and a stable recall of 71.9%. We assume that the high performance of the co-occurrence is mostly determined by the chosen gold-standard. Since all gold-associations were human curated, we assume that most of them are already published, which leads to a bias towards the *Co-occurrence*. The *Grade* yields the lowest precision with 58.4%, but the highest recall of 82%. The *Grade* feature takes the disease and the gene separately into account. We assume that this influences the precision negatively. Since the *Grade* simply counts the number of occurrences in the database, the high recall is maybe given through a low occurrence-threshold. In contrast to this is the *Odds Ratio*, which yields a high precision of 80.7%, but lacks at the recall (28.8%). Its F-Score of 41.8% is the lowest of all feature groups. The *TF-IDF* reaches the highest precision with 91.3%, and is thereby 3.2 points higher than the *Co-occurrence*. The relatively low recall of 65% leads to the second highest F-Score of 75.9%. The performance of the *Path Signature* model reaches an F-Score of 70.3%, which is, in comparison to the other results, on average. The evaluation of the *Connectivity* model results in an F-Score of 65.1% with a precision of 60.8% and a recall of 70.1%, which is a very stable result, too.

A conspicuous behavior can be extracted from these results. We noticed that features, which take the disease and/or the genes separated into account, rather lead to a higher recall. This can be seen in the *Grade*, *Path Signatures* and *Connectivity* feature group. The other four feature groups rather lead to a higher precision.

The comprehensive co-occurrence based model, *CBF*, reaches a very high precision of 91.1% and a moderate recall of 76.6%, which leads to an F-Score of 83.2%. The combination of all co-occurrence based features outperforms each model that is based on a single feature group. However, the addition of gene-connectivity features increases the performance at 1.3 points and is thereby the best performing model in FG. The F-Score of 84.5% is a composition of the slightly lower precision of 89.7 and the 2.8 points higher recall of 79.8%.

Best N Signature Feature Models A model in BSF is determined through the best *N Path Signature*-features (see Section 5.5.2) according to their IG. Based on their IG, four models were built with increasing number of features. Our BSF-experiments started with a model that contains the best 50 *Path Siganture*-features, namely *Best50Signatures*. All other model-names in BSF can be interpreted analogously. We evaluate four models in BSF by performing a ten-fold-cross validation on the training data. The results are presented in Table 6.2.

Model Setup	Precision	Recall	F-Score
Best50Signatures	55.7	84.2	67.1
Best90Signatures	57.0	83.0	67.6
Best140Signatures	57.9	83.3	68.3
Best190Signatures	58.1	82.3	68.1

Table 6.2: Evaluation results of the BSF-classification models on the training data, using a 10-fold cross validation. The highest value for precision, recall and F-Score is highlighted.

The *Best50Signatures*-model reaches an F-Score of 67.1% with a precision of 55.7% and a recall of 84.2%. Taking 90 features into account, shows an increasing precision and a decreasing recall. The model *Best90Signatures* performs with a precision of 55.7% and a recall of 83%, which results in an F-Score of 67.6%. A similar behavior can be seen by increasing the number of features to 140. The precision increases to 62.7%, while the recall drops to 70.3%. The *Best140Signature*-models reaches an F-Score of 66.3%. The last tested model, the *Best190Signatures*, pursues this trend. It reaches the highest precision of all tested models in BSF with 83.2%, but the lowest recall of 57.1%. In all tested models in BSF, we recognized an almost constant F-Score of 67.7%S (+-0.6).

Best M Feature Models A model in BMF contains the best *M* features, according to their IG, regardless of the feature group, which leads to a large overlap to the BSF models. However, in this results, we can see the impact of co-occurrence-based features. The models in BMF are named analogously to the models in BSF. We evaluate them by performing a ten-fold-cross validation on the training data. The results are presented in Table 6.3.

Model Setup	Precision	Recall	F-Score
Best20	91.1	73.8	81.5
Best50	90.8	73.7	81.4
Best100	88.9	74.5	81.1
Best150	87.5	76.9	81.8
Best200	87.5	76.8	81.8

Table 6.3: Evaluation results of classification model on the training data, using a ten-fold cross validation. The highest value for precision, recall and F-score is highlighted.

The first model, *Best20*, reaches a precision of 91.1%, a recall of 73.8% and thereby an F-Score of 81.5%. By adding the next 30 best features, the precision and recall slightly decrease at 0.3 points, respectively 0.1 points. This trend can be pursued in the next three model evaluations. The more features we added, the lower the precision, the higher the recall. However, all evaluated models have an almost constant F-Score of 81.4% (+-0.4 points).

Evaluation on Test Set In Table 6.1, we saw that the co-occurrence based feature rather leads to a high precision. In Table 6.2, we saw that increasing the number of signatures leads to a decreasing precision, while the recall increased. A similar (but inverted) behavior could be recognized in the last experiments in which we took all types of features into account. The result was shown in Table 6.3. All described experiments so far, were done, suing a ten-fold cross validation on the training set.

In our last classification-experiment, we evaluate the best models from the previous experiments on the test set against our baseline model. We choose the pure document based disease-gene co-occurrence as baseline. Thus, our baseline-model contains just a single feature: the co-occurrence as defined in Equation (5.13). We evaluate all models on the test set. The results are presented in Table 6.4.

Setup On Test Set	Precision	Recall	F-Score
Baseline	89.8	57.9	70.4
Best150	87.0	78.1	82.3
Best190Signatures	59.6	81.1	68.7
CBF+Connectivity	86.9	81.9	84.3
CBF	92.1	78.8	85.0

Table 6.4: Evaluation results of the best performing classification-models on the test set in comparison with the baseline-model. The highest value for precision, recall and F-score is highlighted.

Similar to the *Co-occurrence*-model in our first experiments, the baseline is clearly stronger in the precision with 89.8%. However, due to the miss of both other features in *Co-occurrence* (see Section 5.5, second paragraph), the recall stays at 57.9% in the baseline and got an relatively high F-Score of 70.4%. The *Best150* model performs similar to its evaluation on the training set. The precision is at 87.0%, the recall at 78.1% (slightly higher than on the training set) and it has an F-Score of 82.3%. Thereby, this model outperforms the baseline at 12.1 points. The *Best190Signatures*-model lacks in the precision (59.6%), but has a very high recall with 81.1%, which results in 68.7% F-Score and cannot reach the performance of our baseline. The evaluation of the *Co-Connectivity*-model shows very stable results in precision (86.9%) and recall (81.9%). The results are similar to the evaluation on the training data with a slight drop in precision and an increased recall. It reaches a performance of 84.3% in F-Score. The removal of the gene-connectivity features leads to a higher precision and lower recall. The *CBF*-model performs at best with 85.0% in F-Score with a precision of 92.1% and a recall of 78.8%.

Summary By investigating the evaluation results, we can extract several striking results:

1. Features, that are based on the gene-interaction network, rather lead to a higher recall instead of high precision.
2. On the contrary, disease-gene co-occurrence based features rather lead to a higher precision instead of high recall.
3. If a model contains solely *Path Signature*-features, increasing the number of signatures leads to an increasing precision and a decreasing recall.

4. In hybrid models, increasing the number of signature-features leads to a decreasing precision and an increasing recall.
5. The best result on the test set can be achieved by discarding all non-co-occurrence features. We assume that this is mainly given due to the fact that all positive associations from the gold-standard (GTR) were already published, which creates a bias towards co-occurrence based features. If we are interested in mapping the gold-standard most accurately, this is the best model to do it. However, this may not be the best model to find novel disease-gene associations .
6. We were able to outperform the baseline-model at 14.6 points in F-Score.

6.2 Ranking of Gene Candidates

In this section, we present the evaluation results of our experiments, regarding to the task of gene candidate ranking. In the following sub-section, the general methods and experimental setups is described. In Section 6.2.2, we present and discuss the evaluation results. The main objective is to find the best performing model that represents the *DisGeNET*-gold standard with respect to our assumption in Example 6.1.

6.2.1 Methods and Experimental Setup

Ranking with Support Vector Regression In the previous task we used an SVM for binary classification. In the following experiments we need are machine learning method that assigns a numerical value to a new data point. With these regression-values we are able to sort/rank a gene-list. In the following experiments a *Support Vector Regression* (SVR) is used. The SVR is a special case of an SVM and it shares the basic concepts of finding the best fitting hyperplane that maximizes the margin, and minimizing the error. To deal with non-linear separable data, we use the *Gaussian*-kernel function. In addition it has a third parameter called epsilon which can be seen as a margin of tolerance. The performance of an SVR depends on three parameters, the cost-function parameter C, the kernel-parameter *gamma* and the tolerance parameter *epsilon*. In this work a standard ε-SVR is used which was provided by LibSVM [15], integrated into the WEKA environment. For a detailed description we refer to [61] and [15].

Meta Parametrization Likewise to the SVM optimization, the performance of an SVR depends on the cost-function parameter C and the kernel Parameter *gamma*. In addition, a third parameter needs to be optimized, namely the *epsilon* (ε).

- The ε-parameter determines the width of the ε-decision-boundary, which is used to fit the training data. The higher ε is chosen, the less support vectors can be used to build the regression function. Higher ε-values also increase the 'flatness' of the regression function.

The best value-combination of the three parameters can be determined in an advanced grid-search. To do so, we use the WEKA-integrated grid-search method [29], similar to the determination in the classification task.

Feature Reduction For the classification, we estimated the best features by determining the features- information gain. However, the IG can only be calculated for nominal class values. A regression-method assign and uses numeric class values, which denies us to use the IG. Instead, we use the *Correlation Feature Selection* to estimate the best subset of features. The CFS-method [30] determines the best subset with respect to the correlation of the features within the current subset and their correlation to the classification. The higher the features are correlated to the classification, the higher is the worth of the subset. In addition, the higher the correlation between the features in the subset, the lower is the worth of the subset. We use the WEKA-integrated CFS-method to determine the subset with the highest merit.

Measurements for Regression Experiments To measure and compare the results of the gene-ranking evaluation, we use the *Mean Reciprocal Rank* [20,70] (MRR). We determine the *Correlation Coefficient* (CC) and compare the *Mean Absolute Error* (MAE) / *Root Mean Square Error* (RMSE). CC, MAE and RMSE are standard measures of a regression. Thus, we do not further explain them here, but refer to [29, 30, 41] or [61] for more information.

The MRR is an algorithm to measure the quality of a ranking approach according to a set of queries. In contrary to the standard F-Measure, the MRR is calculated with respect to the order of the results. Thereto, it compares the order of a ranking system, which produces a sorted list of elements, to a gold-list of elements. The MRR is always calculated for a set of queries and builds the arithmetic mean of the reciprocal rank as shown in Equation (6.4).

$$MRR = \frac{1}{N} * \sum_{d}^{D} \frac{1}{rank(G_d, r_{d0})} \tag{6.4}$$

N is the number of diseases (queries) to evaluate, calculated as $N = |D|$. G_d is the gold-list of genes (elements) for the specific disease d, and r_{d0} is the first gene in the result-list that was created by the ranking-system for the disease d. The function $rank(G, r)$ returns the index of the element r in the gold-list, if it exists, or zero and is defined as follows:

$$rank(G, r) = \begin{cases} \text{index of } r \text{ in } G & \text{if exists} \\ 0 & \text{else} \end{cases} \tag{6.5}$$

Training and Test Set To train the SVR-models, we need a training and test set, similar to the classification sets. However, they differ in one important fact. While the classification sets contain binary labeled data points, the regression deals with numerical labels. Thus, we choose *DisGeNET* (see Section 3.3) as gold-standard-data. DisGeNET provides a confidence score for each disease-gene association. In order to reduce the number of training and test data points, and to decrease the amount of false positive data points[1], we choose only those associations as positive examples, whose confidence score is greater than 0.1 (see footnote[2]). After extracting all positive training examples from DisGeNET, we extract almost the same amount of negative examples from our database. A disease-gene-association within DGA-DB is a negative training example, if and only if the association is not mentioned in DisGeNET. All negative training examples get the class-value 0. As we did it before, our extracted positive and negative training data are divided into an 80-20 split. We use 80% as training data and 20% as test data. The training data consists of 25.294 data points, split into 11.658 negative and 13.636 positive data points. The test data contains 6.320 data points, split into 2.912 negative and 3.608 positive data points. To ensure comparable results during the evaluation of the different models, we fixed these training and test sets.

[1]The associations in DisGeNET are not human curated but extracted automatically. Thus, the there is no 100% evidence for each association.

[2]The threshold of 0.1 was motivated by the confidence-scoring function of DisGeNET.

6.2.2 Evaluation Results and Discussion

Our experiments on the ranking-models are separated into three parts. At first we evaluate some models on the training set, using a ten-fold-cross validation. During the training, we adjust the meta parameter and do some feature-setting experiments. The goal of the pre-evaluations is to find the best models and settings, which are then tested against the test set. Both steps are similar to the classification-experiments. Due to the combination of the high number of training examples, conducting an experiment is very time consuming. However, our feature-reduction-method CFS aims at optimizing the CC, which is also one of our key-measures. This allows us to reduce the experiments of feature-variations without too much loss of information. We assume that the used CFS-method returns the subset of features that maximizes CC. With respect to the MAE and RMSE, we investigate a few more models on the training set. All results of the evaluated models are shown in Table 6.5.

In the second part, we evaluate the best performing and most interesting models on the test set. The results are shown in Table 6.6. Similar to the first part, we measure the results with respect to the CC, MAE and RMSE. The models are also compared to our baseline-model.

In the third part, we evaluate the same models from part two, but in a different way. Thereto, each of these models (which were previously build on the training data) is integrated into the ranking-system. We then apply all diseases from the test set to the system, which generates a sorted result-list of genes. We compare the result-lists for each disease to the corresponding gold-list, by calculating the reciprocal rank. Finally the MRR can be determined for each model. The evaluation results of these experiments are shown in Table 6.7.

Evaluation on Training Set We evaluate several models on the training data, using a ten-fold-cross validation. The main objective was to find the best meta parameter and the best performing models. We show the best evaluation results in Table 6.5.

The first model that was tested on the training data consists of the best subset, which was generated by the CFS-method. We called this model *CFS328*. Using this feature subset, the model leads to a CC of 0.5663. The performance reaches a MAE of 0.1224 and a RMSE of 0.1426. The high CC does not automatically lead to a low MAE/RMSE. Thus, we evaluate the *Co-occurrence Based Feature*-model (CBF) and the *Co-occurrence based Features + Gene Connectivity Features*-model (CBF+Connectivity). The CBF shows a CC of 0.5480 and a MAE of 0.1213, which is slightly better, and a RMSE of 0.1457, which

Setup On Training Set	CC	MAE	RMSE
CFS328	0.5663	0.1224	0.1426
CBF	0.5480	0.1213	0.1457
CBF+Connectivity	0.5709	0.1146	0.1437
CFS399Sig	0.038	0.1475	0.1814

Table 6.5: Evaluation results of regression model on the training data, using a ten-fold cross validation. The highest value for CC, MAE and RMSE is highlighted.

is slightly worse than the performance of the *CFS328*-model. By adding the *Connectivity*-features, surprisingly, we reach the highest CC of all tested models. It shows 0.5709 points. The MAE shows the lowest error so far with 0.1146. However, the RMSE with 0.1437 is between the *CFS328*-model and the *CBF*-model . Finally we tested the best CFS-model with the restriction to *Path Signature*-features, namely *CFS399-Sig*. It leads to almost no correlation in the data points. The CC is at 0.038 and the performance shows a MAE of 0.1213 and a RMSE of 0.1475 .

Evaluation on Test Set We evaluated some of the previous generated models against the test set and compare our result with our baseline-model. Likewise to the classification-experiments, our baseline-model only takes the pure co-occurrence into account, as defined in Equation (5.13). The results are presented in Table 6.6.

Setup On Test Set	CC	MAE	RMSE
Baseline	0.4358	0.1344	0.1609
CFS328	0.5806	0.1249	0.1453
CBF+Connectivity	0.5831	0.1164	0.1454

Table 6.6: Evaluation results of regression-model on the test data. The highest value for CC, MAE and RMSE is highlighted.

Our baseline-model shows with 0.4358 the lowest CC of all tested models . It performs with an MAE of 0.1344 and an RMSE of 0.1609. The baseline is slightly outperformed by both tested models. The *CFS328*-model shows a data correlation of 0.5806. The MAE drops at 0.0095 to 0.1249. The RMSE drops to 0.1453. The *CBF+Connectivity*-model shows a slightly higher correlation with 0.5831, the lowest MAE of 0.1164 and almost the same RMSE like the CFS328-model of 0.1454.

According to the evaluation results, we assume that gene-ranking is a more difficult or complex task than gene-classification. In the classification experiments, we showed that a C-SVM is able to learn, with an F-Score of over 80%, whether a gene is related to a disease or not. Since the SVR and the SVM share the same basic concepts, we assume that the low results of the ranking experiments are affected by the chosen gold-standard. In contrast to GRT, DisGeNET is generated automatically, using various methods. Thus, it may includes some false positive training examples, which may have affected the correlation of the data negatively, if a correlation exists. In further experiments with less training data, by increasing the confidence threshold, this behavior should be investigated. However, this goes beyond the scope of this thesis and is left for future work.

Evaluation of the Ranking System In our last ranking-experiment, we evaluate our system by integrating the previously described models into our system. For each model, we apply all diseases in the test set to our ranking-system, which creates a ranked list of genes for each disease. We compare the result-lists to the corresponding gold-list, which is created with the positive data points of the test set. The results are shown in Table 6.7.

Model	MRR
Baseline	0.6909
CFS328	0.7067
CBF+Connectivity	0.6816

Table 6.7: Evaluation results of the system by integrating a model.

Integrating our baseline-model leads to a Mean Reciprocal Rank of 0.6909. It is slightly outperformed by the *CFS328*-model with an MRR of 0.7067. Although, the *CBF+Connectivity*-model showed slightly better results in the evaluation on the test set, its MRR shows the lowest result with 0.6816.

Summary By investigating the evaluation results, we noticed the following things:

1. The highest data correlation can be observed by using the *CBF+Connectivity*-model. However, it reaches a CC of just 0.5831. We assume that the low data

correlation is somehow affected due to the chosen gold-standard and its resource generation.

2. The best subset that can be extracted, using only *Path Signature*-features leads to almost no correlation between the data. The CC is at 0.0375. In addition, it had the lowest performance of all tested models with 0.1213 in MAE and 0.1475 in RMSE. By taking all features into account, the best subset shows a slightly better performance of 0.1453 in RMSE. This displays the high influence of co-occurrence based features.

3. We were slightly better than the baseline-model on the test set with a drop in RMSE of 0.0146, and an increased MRR of 0.01. The comparison emphasizes again the high influence of the co-occurrence feature. Likewise to the results in the classification experiments, this can be partly explained, due to the exposure of positive results of disease-gene association results. However, the significance of the improvements needs to be determined in further experiments.

6.3 Case Study with Pulmonary Fibrosis

In the previous both experiments, we evaluated the trained models that should finally be integrated into the system. We determined the best meta parameter and feature selection for the classification and the ranking task. With the information that could be extracted from the previous experiments, we built a comprehensive model, taking all positive and negative data points into account. In this case-study, we integrated the final classification-model into our system. We used *Pulmonary Fibrosis* as testing disease. We applied the disease to our system and generated a file with 200 predicted gene-candidates. These genes are sorted by the number of publications that contain evidence for the disease-gene association. The file was then evaluated by biomedical experts. In the following, we describe the analysis of the generated file.

Pre-evaluation Results of the Pulmonary Fibrosis-Classification Task In a pre-evaluation of the predicted gene-candidates, each of the 200 genes is classified with one of the following labels:

- *True Positive*: Hit was found via co-occurrence.There is a clear connection to *Pulmonary Fibrosis.*
- *Pathway Candidate*: Hints were found, that the gene is associated with *Pulmonary Fibrosis* through relevant mechanism or pathways.
- *Unclear/False Co-occurrence*: Hit was found via co-occurrence, but the evidence for a connection to *Pulmonary Fibrosis* is unclear or negative.
- *False Positive*: False positive in gene recognition.
- *Potential Candidate*: Needs to be investigated by fibrosis experts. Unfortunately, we do not yet have any evaluation results, regarding to the potential gene-candidates.

The results of the pre-classification is shown in Table 6.8.

	True Positive	Pathway Candidate	Unclear/False Co-occurrence	False Positive	Potential Candidate
Count	54	12	26	19	89

Table 6.8: Results of the pre-labeling evaluation for 200 gene-candidates.

True Positive The pre-labeling evaluation shows that we were able to extract 54 true positive genes. These genes are already known and can be extracted from the literature through document-based co-occurrence. A further exploration of the true positives shows that a lot of those genes are also related to cancer-diseases. This is probably caused by two facts:

1. *Pulmonary Fibrosis* and *Cancer* are both tissue proliferation-diseases. Thus, they share some basis symptoms, which may explain the overlap of genes.
2. Cancer-diseases are one of the most extensive studied disorders. This results in a very large proportion of publication that are related to cancer. The high amount of cancer related genes is may be attributable to a publication-bias towards cancer-diseases.

However, the predicted gene-candidates, which are also related to cancer, can be seen as sensible hypothesis. Their actual relations to *Pulmonary Fibrosis* need further laboratory experiments or experimental gene-expression data to prove that the genes are expressed in the lung.

Pathway Candidate On top of that, we were able to find 12 genes that were related to *Pulmonary Fibrosis* through gene-pathways. For example, we predicted the gene *CDC42*, which is related to the gene *PI3K*, using a direct literature-evidence.

Unclear/False Co-occurrence 26 predicted genes are labeled as *Unclear/False Co-occurrence*. One reason to assign this label, was if the provided evidence is based on old publications. In few cases, the considered document contains a negative relation of a gene to *Pulmonary Fibrosis* e.g. the publication with the PubMed ID 22273745:

> „Aryl hydrocarbon receptor (AhR) regulates silica-induced inflammation but not fibrosis.“

In the current system, we make use of a simple document based co-occurrence approach to perform a pre-selection of gene candidates. This method includes no semantical or contextual information at all, which leads to such false positives. However, those false positives are very rare and their elimination demands a benefit-cost analysis in the first place. A third reason to label a gene-candidate with *Unclear/False Co-occurrence*, is a unclear relation to the disease. This could happen, if the gene-candidate is related to a positive gene among longer or unknown pathways. To distinguish between false positives that may have arise from wrong gene-normalization) and complex-true positives, it requires a deeper investigation of the pathway.

False Positive 19 out of 200 are labeled with *False Positive*. Those genes are either a product of a false gene recognition/normalization or belong to a viral-protein. To identify such viral-proteins, it requires deeply contextual information. This information can be extracted from the publication, in which the genes were described as disease-related (directly through co-occurrence or indirectly through gene-pathways). However, the disambiguation of proteins and viral-proteins goes beyond the scope of this thesis.

Potential Candidate All genes that need further investigations to determine, if they are related to *Pulmonary Fibrosis* or not, are labeled with *Potential Candidate*. Our system aims at the prediction of those potential gene-candidates and we were able to extract

89. As mentioned before, the evaluation of these genes is very time consuming and requires biomedical experts for fibrosis-diseases. As yet, we cannot draw any conclusions concerning the accuracy of these candidates.

Summary From the pre-evaluation of the case-study, we summarize key-points:

1. We applied *Pulmonary Fibrosis* to our System and generate a file that contains 200 predictions of related genes.
2. In this case-study, the *Best150*-classification model was used. It was previously described in Section 6.1.2.
3. The aim of the system in combination with this model lays on the generation of novel genes, that can be related to *Pulmonary Fibrosis*.
4. The genes within the generated files are ranked by their number of involved publications.
5. Our system produced 54 true positives, 19 false negatives, 12 likely-related genes, 26 unlikely-related genes and 89 other gene candidates.

7 Conclusion

In this chapter, we conclude our work. We discuss briefly the systems architecture and point out some advantages and issues that we left for future work. With a higher accuracy, we discuss the evaluation results with respect to the used methods. Here too, we share our ideas for future work in order to improve the systems performance.

The main objective of this thesis was to develop an approach that extracts and ranks novel disease-related genes by their importance. Our work was motivated by the fact, that current systems, which generate such novel gene-candidates, often lack in their ability to deal with the huge amount of produced candidates. In order to achieve this goal, we built a comprehensive disease-gene association extraction system. The approach, we presented in this thesis, tackles the problem of association extraction by aggregating information across millions of scientific publications and uses machine learning methods to learn their importance to a given disease. Thereto, we extended the feature space of common entity-relation features, like disease-gene co-occurrences, by novel features based on a comprehensive gene-interaction network. This network was also utilized to predict novel gene candidates.

A great challenge was to design a comprehensive system that can deal with the large amount of data of publications and extracted information. Besides the very time consuming information extraction of diseases, genes and gene-events, we had to store all information persistently. We reduced offline and online processing time and focused our results to human genes by limiting the taxonomic scope. Further, the design also includes the integration of existing state-of-the-art systems and self developed methods. It implement their interactions properly and puts them into a running and functional pipeline.

In the following, we discuss some key-designs and present ideas for future work. Our system consists of several independent approaches for information extraction. We integrated a CRF tagger by Klinger et al. [36] that tackles the disease-recognition. After an extensive investigation of the recognized disease-mentions, we realized the need of a disease-normalization. We tried to integrate the current state-of-the art framework

DNorm. Using an existing framework, which is described in the literature can be a challenging task. A proper integration into a running pipeline often requires adaptations of the source code and/or a communication with the developer. Both are very time consuming and the outcome is often unpredictable. Due to time issues, we discontinued our efforts and implemented a simple disease normalization framework, called *DiNo*. DiNo's method was motivated by DNorm and with a strict adaptation to the output of the CRF tagger, we achieved an F-Score of 61.49% on the BioCreativeII disease normalization task. Unfortunately, we could not range on the same level as DNorm and there is a lot of room for improvements. In the future, we could either continue our work of integrating DNorm or improve DiNo.

Our combination of TEES and GeNo shows, that the combination of two well performing systems does not necessarily result in an equally performing comprehensive system. We integrated TEES for gene-event recognition, which aims at the extraction of gene-events but does not distinguish between genes of various species. We restricted our system to human genes and integrated the highly performing framework GeNo for the task of gene normalization and disambiguation of the taxonomic scopes. Although, GeNo has an integrated gene recognition module, thus we had to change its pipeline by integrating TEES, since we were mainly interested in event-involved genes. Both facts, TEES' non-limitation to human genes and intersecting GeNo's pipeline, effects the evaluation of the gene normalization negatively. We achieved an F-Score of 55.33% on the BioCreativeII gene normalization task. However, to the best of our knowledge, there is no existing approach that tackles the event extraction with respect to human genes solely. To improve the gene normalization, a future task could be integrating GeNo as gene recognition framework into TEES, instead of the other way around.

In this work, we presented a method to reduce the complexity of gene-events. This method includes several graph transformations, event simplifications and event-type generalizations. In agreement with biomedical experts, we developed a reduced GENIA event scheme, which describes the new modified events. These modifications were motivated in order to increase the gene-interaction density and to reduce the dimensionality of the feature space. Up to this point, we did not investigate the impact of such generalizations, neither run any experiments to ensure their quality. An evaluation of the event-modifications would go beyond the scope of this thesis and is left for future work.

In the following, we discuss the evaluation results of our final experiments. In order to evaluate our system, several experiments were performed that can be divided into three groups. In the first bunch of experiments, we evaluate some classification models. These models were trained with a support vector machine in order to classify a gene as disease-related or not. Our dataset was composed of positive examples that could be gained from the *Genetic Testing Registry* and negative examples from our database. Each data point consisted exactly of one disease, one gene and a label (1 for a positive example, respectively -1 for a negative example). A comprehensive feature vector was created for each data point. To determine the best features, the *Information Gain*-method was used. Our evaluations on the test set comprised several combinations of different features, such as a limitation to the best n, only co-occurrence-based, only signature-based, etc. It turned out, that using the co-occurrence based features led to the highest performance (F-Score of 85.0%). The addition of the *Gene Connectivity*-features led to a drop in F-Score of 0.7 points.

We recognized an interesting behavior by investigating the signature-based models on the training set. By increasing the number of features (signatures), we recognized an increasing precision by simultaneously decreasing recall and an almost constant F-Score of 66.5%. In combination with co-occurrence features, an opposite effect could be recognized. Increasing the number of signature-features led to a drop in the precision but slightly increase the recall. However, it seemed that this behavior converged at a number between 150-200 features. On the test set, the best model reached an F-Score of 82.3%.

We compared all results with the performance of our baseline-model which reached a performance of 70.4% (F-Score) on the test set. A comparison of the results showed that, considering the *Path Signature*-features only, could not outperform the baseline. However, each hybrid-model outperformed the baseline on the training set by at least 9.8 points in F-Score.

We decided to not choose the best performing model (*Co+Connectivity*) for the case-study, but to use the best performing model that includes also signature-features. This was mostly motivated by the following reasons:

1. The positive training examples from the gold-standard are human curated results of experiments. We thereby assume that most of them were already published, which lays a bias towards co-occurrence-based features.

2. One research question of our work was to determine, if signatures contain relevant information to increase the performance of gene classification and gene ranking.

In a second evaluation, we have done several experiments to examine the performance of the ranking task. Therefore, we trained support vector regression models with similar setups to the classification experiments. Positive data points were extracted from the *DisGeNET*-database, which provides a confidence score for each disease-gene association. Likewise to the classification data, we added negative training examples from our database. Instead of determining the *Information Gain*, which can not be used on numeric class labels, we used a *Subset Evaluation*-technique to extract the best feature subset. We compared the best feature-subset model with our baseline model, which got a correlation coefficient (CC) of 0.436 and a *Mean Absolute Error* (MAE) of 0.134. The comparison showed a slight improvement in MAE of 0.01 and a CC of 0.580. However, the best performing model was a hybrid model that uses co-occurrence based features in combination with gene-connectivity features. It outperforms the baseline in CC with 0.583 and a MAE of 0.116. We then evaluated all three models by integrating them into the system and measure the *Mean Reciprocal Rank* (MRR) of the test set. It turned out, that the best model performs at best with a MRR of 0.7067.

In a last evaluation, we conducted a case-study. We applied *Pulmonary Fibrosis* to our system and generated a file that contains 200 predicted gene candidates, using a classification model. In a pre-evaluation, these genes were labeled as true positive, false positive, unlikely related, likely related or potential candidate. We were able to show that our system successfully generates way more true positives and potential candidates than false positives. However, due to the strict timeframe of this thesis, up to this point, we do not have any evaluation results of the potential candidates.

In this thesis, we built a comprehensive system for the task of disease-gene-association extraction across millions of biomedical publications. The aim of this thesis was to integrate a machine learning based approach that uses novel features to learn the importance of genes to a specific disease. Although we recognized that co-occurrence based feature lead to a higher performance, we could show that using the novel features can be used to predict novel gene-candidates. The current system can be seen as an essential foundation that can be further optimized, extended or adapted to similar tasks.

Bibliography

[1] U.S. NLM PubMed. `http://www.ncbi.nlm.nih.gov/pubmed`. Accessed: 2015-04-25.

[2] Hisham Al-Mubaid and Rajit K Singh. A new text mining approach for finding protein-to-disease associations. *American Journal of Biochemistry and Biotechnology*, 1(3):145, 2005.

[3] Rie Kubota Ando. Biocreative ii gene mention tagging system at ibm watson. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, volume 23, pages 101–103. Centro Nacional de Investigaciones Oncologicas (CNIO) Madrid, Spain, 2007.

[4] Cecilia N Arighi, Zhiyong Lu, Martin Krallinger, Kevin B Cohen, W John Wilbur, Alfonso Valencia, Lynette Hirschman, and Cathy H Wu. Overview of the biocreative iii workshop. *BMC bioinformatics*, 12(Suppl 8):S1, 2011.

[5] Ken Arnold, James Gosling, David Holmes, and David Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 1996.

[6] R Baud et al. Improving literature based discovery support by genetic knowledge integration. *The New Navagators: From Professionals to Patients*, 95:68, 2003.

[7] Jari Björne, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. Drug-drug interaction extraction from biomedical texts with svm and rls classifiers. *Proceedings of DDIExtraction-2011 challenge task*, pages 35–42, 2011.

[8] Jari Björne, Filip Ginter, Sampo Pyysalo, Jun'ichi Tsujii, and Tapio Salakoski. Complex event extraction at pubmed scale. *Bioinformatics*, 26(12):i382–i390, 2010.

[9] Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 10–18. Association for Computational Linguistics, 2009.

[10] Jari Björne and Tapio Salakoski. Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 183–191. Association for Computational Linguistics, 2011.

[11] Jari Björne, Sofie Van Landeghem, Sampo Pyysalo, Tomoko Ohta, Filip Ginter, Yves Van de Peer, Sophia Ananiadou, and Tapio Salakoski. Pubmed-scale event extraction for post-translational modifications, epigenetics and protein structural relations. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 82–90. Association for Computational Linguistics, 2012.

[12] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[13] Dan Brickley and Ramanathan V Guha. Resource description framework (rdf) schema specification 1.0: W3c candidate recommendation 27 march 2000. 2000.

[14] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *The Semantic Web—ISWC 2002*, pages 54–68. Springer, 2002.

[15] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[16] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

[17] Hong-Woo Chun, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, and Jun'ichi Tsujii. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. In *Pacific Symposium on Biocomputing*, volume 11, pages 4–15, 2006.

[18] UniProt Consortium et al. Update on activities at the universal protein resource (uniprot) in 2013. *Nucleic acids research*, 41(D1):D43–D47, 2013.

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

[20] Nick Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer, 2009.

[21] James A Cuff and Geoffrey J Barton. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 40(3):502–511, 2000.

[22] Allan Peter Davis, Cynthia Grondin Murphy, Robin Johnson, Jean M Lay, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Michael C Rosenstein, Thomas C Wiegers, et al. The comparative toxicogenomics database: update 2013. *Nucleic acids research*, page gks994, 2012.

[23] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

[24] Andreas Doms and Michael Schroeder. Gopubmed: exploring pubmed with the gene ontology. *Nucleic acids research*, 33(suppl 2):W783–W786, 2005.

[25] Haw-ren Fang, Kevin Murphy, Yang Jin, Jessica S. Kim, and Peter S. White. Human gene name normalization using text matching with automatically extracted synonym dictionaries. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, BioNLP '06, pages 41–48, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[26] Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[27] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL*, volume 18, pages 401–408. Citeseer, 2006.

[28] Jörg Hakenberg, Conrad Plake, Robert Leaman, Michael Schroeder, and Graciela Gonzalez. Inter-species normalization of gene mentions with gnat. *Bioinformatics*, 24(16):i126–i132, 2008.

[29] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[30] Mark A Hall. *Correlation-based feature selection for machine learning.* PhD thesis, The University of Waikato, 1999.

[31] Ada Hamosh, Alan F Scott, Joanna S Amberger, Carol A Bocchini, and Victor A McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl 1):D514–D517, 2005.

[32] Dimitar Hristovski, Thomas Rindflesch, and Borut Peterlin. Using literature-based discovery to identify novel therapeutic approaches. *Cardiovascular & Hematological Agents in Medicinal Chemistry (Formerly Current Medicinal Chemistry-*

Cardiovascular & Hematological Agents), 11(1):14–24, 2013.

[33] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics, 2009.

[34] Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10, 2008.

[35] Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6. Association for Computational Linguistics, 2011.

[36] Roman Klinger, Christoph M Friedrich, Juliane Fluck, and Martin Hofmann-Apitius. Named entity recognition with combinations of conditional random fields. In *Proceedings of the second biocreative challenge evaluation workshop*, 2007.

[37] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[38] Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, page btt474, 2013.

[39] Robert Leaman, Graciela Gonzalez, et al. Banner: an executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*, volume 13, pages 652–663. Citeseer, 2008.

[40] Robert Leaman, Christopher Miller, and G Gonzalez. Enabling recognition of diseases in biomedical text with machine learning: corpus and benchmark. In *Proceedings of the 2009 Symposium on Languages in Biology and Medicine*, volume 82, 2009.

[41] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

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

[43] Dingcheng Li, Karin Kipper-Schuler, and Guergana Savova. Conditional random fields and support vector machines for disorder named entity recognition in clinical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 94–95, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[44] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[45] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic acids research*, 39(suppl 1):D52–D57, 2011.

[46] Alexander A Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, et al. Overview of biocreative ii gene normalization. *Genome biology*, 9(Suppl 2):S3, 2008.

[47] Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, 2013.

[48] Tomoko Ohta, Yoshimasa Tsuruoka, Jumpei Takeuchi, Jin-Dong Kim, Yusuke Miyao, Akane Yakushiji, Kazuhiro Yoshida, Yuka Tateisi, Takashi Ninomiya, Katsuya Masuda, et al. An intelligent search engine and gui-based efficient medline search tool based on deep syntactic parsing. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 17–20. Association for Computational Linguistics, 2006.

[49] Arzucan Özgür, Thuy Vu, Güneş Erkan, and Dragomir R Radev. Identifying gene-disease associations using centrality on a literature mined gene-interaction network. *Bioinformatics*, 24(13):i277–i285, 2008.

[50] Benjamin Paassen, Andreas Stöckel, Raphael Dickfelder, Jan Philip Göpfert, Nicole Brazda, Tarek Kirchhoffer, Hans Werner Müller, Roman Klinger, Matthias Hartung, and Philipp Cimiano. Ontology-based extraction of structured information from publications on preclinical experiments for spinal cord injury treatments. In *Third Workshop on Semantic Web and Information Extraction (SWAIE). The 25th International Conference on Computational Linguistics (COLING)*, 2014.

[51] Janet Piñero, Núria Queralt-Rosinach, Àlex Bravo, Jordi Deu-Pons, Anna Bauer-Mehren, Martin Baron, Ferran Sanz, and Laura I Furlong. Disgenet: a discovery platform for the dynamical exploration of human diseases and their genes. *Database*, 2015:bav028, 2015.

[52] Sune Pletscher-Frankild, Albert Pallejà, Kalliopi Tsafou, Janos X Binder, and Lars Juhl Jensen. Diseases: Text mining and data integration of disease–gene associations. *Methods*, 2014.

[53] Changqin Quan and Fuji Ren. Gene–disease association extraction by text mining and network analysis. In *Proceedings of the 5th International Workshop on Health Text Mining and Information Analysis (Louhi)@ EACL*, pages 54–63, 2014.

[54] Wendy S Rubinstein, Donna R Maglott, Jennifer M Lee, Brandi L Kattman, Adriana J Malheiro, Michael Ovetsky, Vichet Hem, Viatcheslav Gorelenkov, Guangfeng Song, Craig Wallin, et al. The nih genetic testing registry: a new, centralized database of genetic tests to enable access to comprehensive information and improve transparency. *Nucleic acids research*, page gks1173, 2012.

[55] Rune Sætre, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yuichiro Matsubayashi, and Tomoko Ohta. Akane system: protein-protein interaction pairs in biocreative2 challenge, ppi-ips subtask. In *Proceedings of the Second BioCreative Challenge Workshop*, pages 209–212, 2007.

[56] Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic acids research*, 39(suppl 1):D38–D51, 2011.

[57] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[58] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[59] Mir S Siadaty, Jianfen Shu, and William A Knaus. Relemed: sentence-level search engine with relevance score for the medline database of biomedical articles. *BMC medical informatics and decision making*, 7(1):1, 2007.

[60] S Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[61] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[62] Pontus Stenetorp, Goran Topić, Sampo Pyysalo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. Bionlp shared task 2011: Supporting resources. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 112–120. Association for Computational Linguistics, 2011.

[63] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[64] Don R Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, 1986.

[65] Don R Swanson and Neil R Smalheiser. Undiscovered public knowledge: A ten-year update. In *KDD*, pages 295–298, 1996.

[66] Tzong-Han Tsai, Shih-Hung Wu, and Wen-Lian Hsu. Exploitation of linguistic features using a crf-based biomedical named entity recognizer. In *Proceedings of BioLINK*, volume 2005, 2005.

[67] Yoshimasa Tsuruoka, John McNaught, and Sophia Ananiadou. Normalizing biomedical terms by minimizing ambiguity and variability. *BMC bioinformatics*, 9(Suppl 3):S2, 2008.

[68] Marc A van Driel, Koen Cuelenaere, Patrick PCW Kemmeren, Jack AM Leunissen, and Han G Brunner. A new web-based data mining tool for the identification of candidate genes for human genetic disorders. *European Journal of Human Genetics*, 11(1):57–63, 2003.

[69] Sofie Van Landeghem, Jari Björne, Chih-Hsuan Wei, Kai Hakala, Sampo Pyysalo, Sophia Ananiadou, Hung-Yu Kao, Zhiyong Lu, Tapio Salakoski, Yves Van de Peer, et al. Large-scale event extraction from literature with multi-level gene normalization. *PloS one*, 8(4):e55814, 2013.

[70] Ellen M Voorhees et al. The trec-8 question answering track report. In *TREC*, volume 99, pages 77–82, 1999.

[71] Chih-Hsuan Wei and Hung-Yu Kao. Cross-species gene normalization by species inference. *BMC bioinformatics*, 12(Suppl 8):S5, 2011.

[72] Joachim Wermter, Katrin Tomanek, and Udo Hahn. High-performance gene name normalization with geno. *Bioinformatics*, 25(6):815–821, 2009.

[73] Jonathan D Wren, Raffi Bekeredjian, Jelena A Stewart, Ralph V Shohet, and Harold R Garner. Knowledge discovery by automated identification and ranking of implicit relationships. *Bioinformatics*, 20(3):389–398, 2004.

Eigenständigkeitserklärung

Master's Thesis: Declaration of Authorship - Hendrik ter Horst

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig verfasst und gelieferte Datensätze, Zeichnungen, Skizzen und graphische Darstellungen selbständig erstellt habe. Ich habe keine anderen Quellen als die angegebenen benutzt und habe die Stellen der Arbeit, die anderen Werken entnommen sind - einschl. verwendeter Tabellen und Abbildungen - in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Bielefeld, Mai 2015

Hendrik ter Horst