Experiences from Developing the Domain-Specific Entity Search Engine GeneView

Philippe Thomas, Johannes Starlinger, Ulf Leser

Humboldt-Universitätzu Berlin
Unter den Linden 6, 10099 Berlin
{thomas,starlin,leser}leser@informatik.hu-berlin.de

Abstract: .GeneView is a semantic search engine for the Life Sciences. Unlike traditional search engines, GeneView searches indexed documents not only at the textual (syntactic) level, but analyzes texts upon import to recognize and properly handle biomedical entities, relationships between those entities, and the structure of documents. This allows for a number of advanced features required to work effectively with scientific texts, such as precise search despite large numbers of synonyms and homonyms, entity disambiguation, ranking of documents by entity content, linking to structured knowledge about entities, user-friendly highlighting of recognized entities etc. As of now, GeneView indexes approximately ~21,4 million abstracts and ~358.000 full texts with more than 200 Million entities of 11 different types and more than 100,000 relationships of three different types. In this paper, we describe the architecture underlying the system with a focus on the complex pipeline of advanced NLP and information extraction tools necessary for achieving the above functionality. We also discuss open challenges in developing and maintaining a semantic search engine over a large (though not web-scale) corpus.

1. Introduction

The vast majority of novel findings in Life Science research are first presented in the scientific literature. Over the years, the amount of texts in this domain has grown enormously and has reached a point where finding specific information becomes troublesome. In 2011 alone, MEDLINE archived more than 800,000 new articles, which corresponds to an increase of more than one article per minute. Besides the rapidly growing sheer number of articles, also the length of available texts is growing, as more and more articles become freely available as full text.

Simple and fast access to the scientific literature is enormously important for researchers to keep up-to-date with their field. In the life sciences, researchers typically (but not always) search for information about some specific biomedical entity, like genes,

diseases, mutations etc. Such a search is very difficult for a number of reasons, which we explain using genes as an example. Firstly, genes usually have many synonyms: on average, Entrez gene gives 2.2 synonyms for each human gene, with a maximum of 31 synonyms for the gene OR4H6P (Entrez gene Id 26322). In addition to synonyms, morphological variations are very frequent in scientific articles (e.g. BRCA1 or BRCA-1).Secondly, gene names are highly ambiguous, both with other genes or other biological entities (like diseases), and with common English words. For instance, many genes are named after the phenotype they are associated with, leading to names such as „white“ or „hedgehog“. On the other hand, evolutionary related genes in different species often have the same name although they should be considered as different entities in most applications. Thirdly, single genes are studied from very different viewpoints, often leading to the invention of slight variations of names (like the mRNA created from a gene being named slightly different than the gene itself). Which of these variations are relevant for a given search is difficult to express. And finally, gene names follow no regular structure but can appear as anything from a three letter acronym to a multi-token complex name, which makes spotting a gene name in a search result very hard. Similar problems also exist for many other biologically relevant entity types, such as diseases (whose names often contain ordinary persons' names, like „Wilsons disease“), or medical symptoms (whose names can be used in many different contexts not related to diseases, e.g. “shiver” or “cold”). The situation becomes worse when not only information about a single entity is searched, but about relations between entities, like genes associated to a disease or mutations associated with metabolization rates of a drug. Finding all genes that are targeted by a given drug X is simply impossible with conventional technology, as one cannot express a query like „X and any gene“.

As a result, searches often lead to unsatisfactory results. For instance, [DMNL09] reported that over one third of all 58 Million PubMed queries collected for March 2008 result in hundreds or even thousands of results. It also directly impedes research: [OW04] pointed out that ambiguous nomenclature led to multiple discoveries of the same mutation. Consequently, there is a growing body of research trying to provide improved search for scientific texts [Lu11]. Researchers need to be able to search for entities instead of keywords; they must be allowed to use any of the existing names for an entity; systems should automatically disambiguate homonyms and link entity names to structured information in external databases; search engines should be able to rank search results using different metrics, including consideration of the entities present in a match, etc.

A pre-requisite for such features is the high-quality recognition of entities (also called named entity recognition, NER [LH05]) and relationships between entities in a given text (also called relationship extraction, RE [Sar08]). This area has seen intensive research over the last decade [ZDF+08]. In contrast to other domains, where especially

NER seems to be considered as an essentially solved problem [Bal12], in biomedicine both problems are far from having been solved in a satisfying manner. For instance, the best gene recognition systems to-date achieve an F-measure of roughly 85% [KMS+08]; the best chemical taggers reach less than 70% F-Measure [RWL12]; the best tools for recognizing disease names reach around 80% F-measure [CL10]. The situation is worse when it comes to RE. The currently best systems for recognizing drug-drug interactions reach an F-Measure of roughly 65% [TNS+11]; recognition of protein-protein-interactions, despite that literally hundreds of papers have been devoted to this topic, still cannot be performed with more than ~60% F-measure [BKS10]. Clearly, problems in RE are highly correlated to problems in NER. For instance, if a protein recognition tool reaches an accuracy of 80%, recognition of relationships between two proteins can hardly be performed any better than at 64% accuracy (assuming that difficulties in recognizing a protein's name and difficulties in recognizing whether this and another protein interact are not correlated).

Tools that are state-of-the-art in such tasks usually are the result of long-term research and encompass considerable amounts of experience, effort, and time. Many of them are freely available. However, implementations differ in terms of programming language, required libraries, dependencies from other tools, configuration etc. Especially the dependency of NER and RE methods on textual preprocessing with specific NLP tools sometimes makes it necessary to process the same text multiple times with essentially the same goal (like POS tagging), but using different tools. Building a high-quality entity search engine thus requires bundling the best available algorithms into complex pipelines of different algorithms processing the same text with a different purpose. Each algorithm produces specific annotations, which often need to be transformed into different formats to be read by the next algorithms.

In this paper, we describe GeneView, a full-fledged entity search engine for biomedical publications. It currently identifies and normalizes ten different entity types (chemicals, cell-types, diseases, drugs, enzymes, genes, histone modifications, single nucleotide polymorphisms (SNP), species, and tissues) and three relationship types (protein-protein-interactions, regulatory relationships, and drug-drug-interactions[1]) and indexes app. 21.4 million abstracts and almost 360.000 full texts. The amount of structured information it generates and makes available to its users is vast; altogether, we extracted more than 210 million entity mentions and more than 8,3 Million relationships [TSV+12]. Compared to other entity search engines in the field, it is either more complete in terms of coverage of entities/relationships or provides information of higher quality (and in most cases both). For instance, our previous system Alibaba had a similar coverage, but performed NER using dictionaries and RE using co-occurrence, both of

[1] Note that not all recognized relationships are displayed at the web interface yet.

which achieve suboptimal results. The system probably most similar to GeneView from an IE point-of-view is BioContext [GSBN12], which indexes only three different entity types. Furthermore, it only performs the IE and is not integrated into a search engine. GeneView has a number of features, which to our knowledge are not available in any other (biomedical) search engine. For instance, we support ranking of search results by entity counts. A user interested in mutations of a specific gene may search for this gene and then ranks the (probably many) results by the number of mutations they describe. Another unique feature is personalized ranking; therein, users may define their own gene lists and use the number of occurrences of genes from this list as ranking criterion for search results.

A general overview of GeneView including an intensive discussion of biological applications has been published elsewhere [TSV+12]. In this paper, we focus on the engineering challenges one faces trying to build a search engine of the coverage, quality, and depth of GeneView. We believe that these challenges are similar also in other domains and thus hope that sharing our experiences might prove useful for many other researchers. A specific intention of this paper is to re-emphasize the complexity of high-quality information extraction in many domains, in contrast to many recent works which essentially consider IE problems (in their domain) to be solved and focus on merging, using, or querying extracted information. The remainder of this paper is structured as follows. In the next chapter, we describe the user interface of GeneView and sketch some applications. Chapter 3 explains the architecture behind GeneView and will especially focus on the IE pipeline used to drive it. In Chapter 4, we focus on how extracted results are used for ranking and display during search. Chapter 5 concludes the paper and summarizes a number of lessons learned from building the system.

2. User Interface

GeneView provides a user-friendly web-interface to make the extracted entity data searchable and accessible (see Fig. 1). GeneViews search bar, which is provided at the top of every page, allows users to issue keyword queries on all available text documents. This includes entity-specific search for recognized entities using standard identifiers, e.g., Entrez gene ids for gene identification. The search bar offers an auto-completion function to make it easier to find specific identifiers. For instance, typing BRCA1 into the search bar will bring up suggestions for several, species-dependent Entrez gene identifiers this short gene name corresponds to. To provide this functionality, GeneView uses a dictionary containing all entity mentions found in PubMed, each associated with their corresponding identifier. Additionally, the search form provides various options for result ranking and filtering. For instance, the user can choose to only include publications in the search result, which have been found to include certain types of

entities (e.g., genes, SNPs, or chemicals). Figure 1 shows the result listing for a search for publications containing two specific genes identified by their Entrez gene id. The result is sorted by date of publication and has been filtered to only contain articles that also contain at least one SNP.

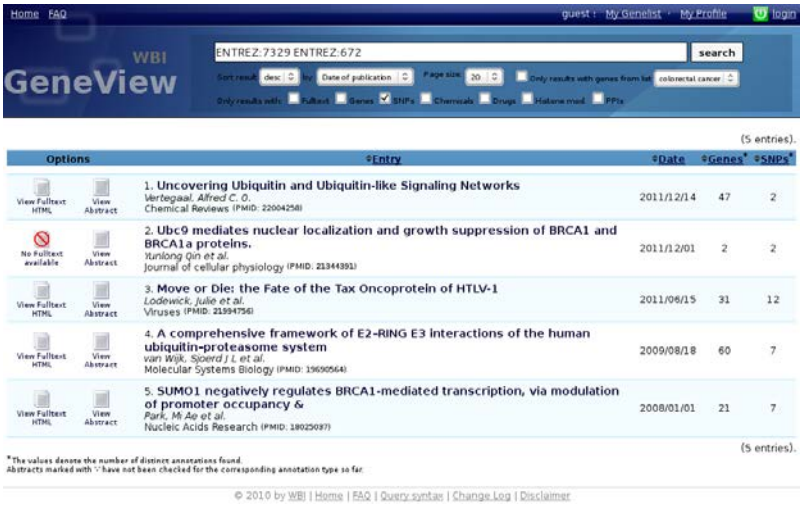

Figure 1: Result of a search for texts mentioning two specific genes, filtered for SNP content, sorted by date of publication.

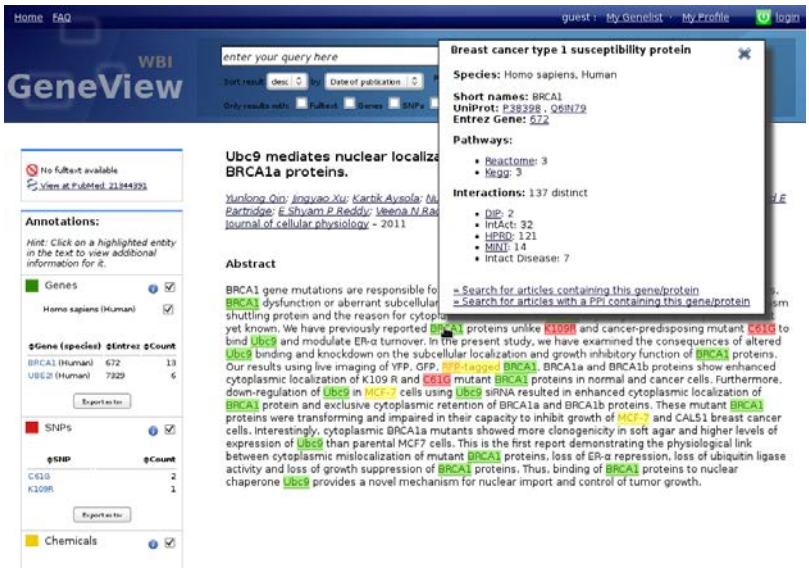

Figure 2: GeneViews single article view of PubMed ID abstract 21344391. Inline entity highlighting is complemented by an overview of entities found in the text (left-hand bar). Highlighted entities provide pop-ups with additional information from external databases.

Clicking on a search result shows the selected article together with all annotations (see Fig. 2). Recognized entities are visualized by type-specific color highlighting. All entities are clickable to provide additional information such as link-outs to external reference databases. These pop-ups also provide links to search for content related to the selected entity. GeneView also provides an overview of all entities found in the article (Fig. 2, left-hand bar). This is particularly helpful when dealing with full text papers containing multiple mentions for various entity types.

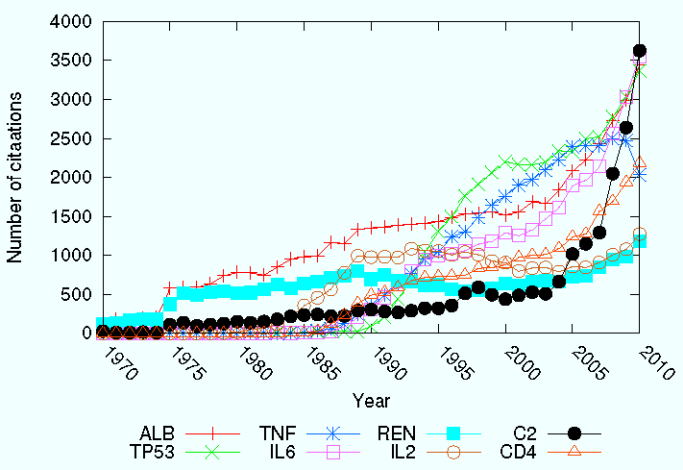

Figure 3. Number of citations mentioning one of the eight most frequently occurring genes over the last 40 years.

The above example of SNP-filtered searching for specific genes demonstrates one important use case of GeneView: The ability to use information about several types of biological entities in a single query both for ranking and for defining what constitutes the primary search result. With the given example, a user can easily retrieve all publications, which mention a mutation in the context of the given gene of interest. GeneView makes such complex cross-entity searches a convenience. While GeneView extracts information about several types of entities to enable this type of multi-entity search, it does have special support for genes/proteins. The on-click information available for genes is currently the most comprehensive one, containing links to several external reference databases and providing information on pathways and protein-protein interactions the gene/protein participates in. The pop-up also provides the option to search GeneView for articles describing PPIs in which the given gene/protein is found.

Besides the user interface, the information backing GeneView enables many interesting types of analysis in its own right. These include searching for trends in appearance of entities, as displayed in Figure 3 showing the citation counts over the last 40 years for the 8 most often recognized genes. The temporal information for a mention is taken from

the publication data of the article containing it. This functionality will, in the future, also be available for sets of genes, in particular pathways, and will be presented in the form of movies showing how information on complex pathways has grown over time.

3. Architecture and Pipeline

GeneView indexes all available articles from PubMed and PubMed Centrals open access set. Together with each articles text we store metadata such as authors, journal, MeSH terms, and figure/table captions that can be extracted by XML parsing from the original NCBI files. All texts are imported into Lucene[2], serving as storage, query, and ranking engine. Metadata and information about all recognized entities, especially type and Id of the entity and the exact position in the text, are stored in a relational database to allow structured retrieval (see Figure 4). Upon import, texts are processed by a custom text-mining pipeline that incorporates a multitude of tools for pre- and post-processing and for the entity-specific steps of NER, NEN and RE (see below). We decided not to use frameworks like UIMA, as most of our incorporated tools are not provided as UIMA components and would have required developing a proper wrapper. Furthermore, testing components inside of UIMA is, in our experience, extremely difficult.

Physically, the Lucene index, the web-server, and the database are located on three different machines. The web-server communicates with the Lucene index using an XML-RPC interface. User defined queries are sent to this index together with pagination and ranking preferences, where requests are processed and results are filtered and ranked. Values of user-selected check boxes (like “show only results with SNPs”) are converted to appropriate Lucene query options and appended to the query. We explain details of this process in the rest of this chapter.

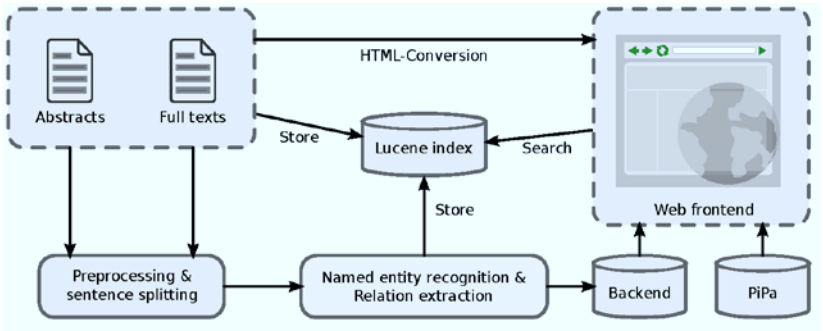

Figure 4. Architecture of GeneView.

[2] See http://lucene.apache.org/core/

Document preprocessing

All texts are downloaded from the National Library of Medicine (NLM) as XML. Available full text articles are converted into HTML for display in the GeneView web interface using XSLT scripts provided by NLM[3]. This transformation generates HTML representations resembling the PubMed Central visualization and thus enables a similar user experience. During this conversion, HTML specific characters like "&amp" are replaced with the corresponding UTF-8 symbol. HTML elements (e.g. </p> or <body>) are ignored and references at the end of the document are removed. Similarly, HTML tables are ignored. This conversion is necessary, as all text-mining steps require such clean text; in effect, we need to store each text twice, once for web display, and once for internal processing. This duplication seems to be inevitable, but generates additional problems when it comes to exactly addressing text snippets for syntax highlighting. Essentially, we need to maintain an exhaustive mapping from the cleansed text back to the HTML file. For articles without full text, i.e., usually PubMed abstracts, HTML is generated on the fly from the information stored in the Lucene index. Before starting the core information extraction pipeline, we detect sentence boundaries, section names, and abbreviations/long form mappings using the algorithm from [SH03]. Section names are identified using an approximate dictionary covering the 200 most often occurring section names[4]. This allows us to recognize 99.7% of all occurring section headings. We use this information for weighting search terms differently depending on the section of a document they appear in, a method which has proven highly effective in several works [DWH10; HRL05].

Named Entity Recognition and PPI extraction

The pre-processed texts are piped through a series of NER and RE tools (see Figure 5). These tools were selected using a best-of-breed strategy; some of them were developed in house, some are external. We do not discuss those tools in detail here but refer to the original publications. The most important ones are (1) GNAT for gene and protein names [HGH+11], (2) MutationFinder for detecting SNPs [CBR+07], ChemSpot for chemicals [RWL12], and (4) Linnaeus for species names [GNB10]. Most of these tools use mixtures of machine learning algorithms (mostly Conditional Random Fields) trained on gold standard corpora and exhaustive dictionaries of the respective entity type.

The next step in the pipeline is relationship extraction. For this purpose, we use the freely available framework by Tikk et al. [TTP+10] which combines necessary NLP

[3] ftp://ftp.ncbi.nih.gov/pub/archive_dtd/archiving/

[4]Note that section names in biomedical papers, in contrast to computer science, are highly standardized.

tools and a set of 14 different kernel-based RE methods. Of those, we use the two best performing algorithms (according to [TTP+10]), i.e., APG [APB+08] and SL [GLR06]. SL uses a SVM for classifying pairs of entities found in a sentence based on large bag-of-word-style feature vectors of the text surrounding the entities. APG applies a similar method, but uses a far larger vector including features derived from the dependency parse trees of the sentences. Therefore, sentences have to be parsed prior to the application of APG.

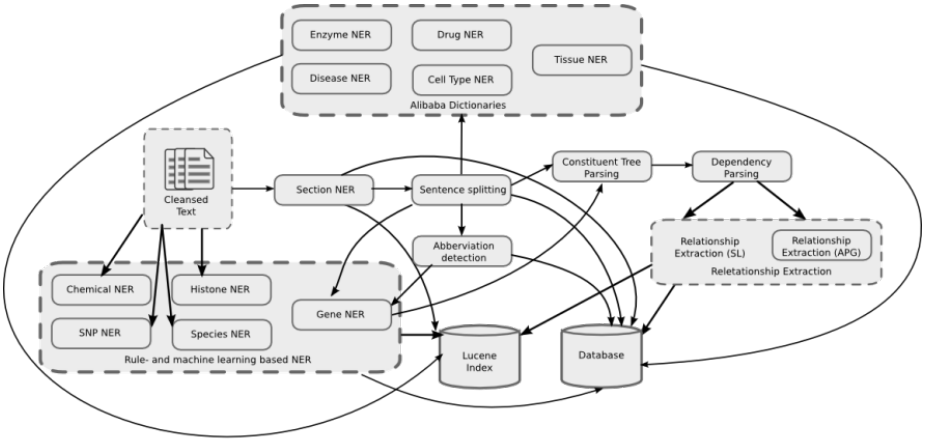

Figure 5. Pipeline of information extraction and NLP tools for creating the GeneView index.

A persistent problem with using tools developed independently is that they require different input. Tools may require tokenized text, or may depend on unprocessed text because they perform their own tokenization. Similarly, some tools require text to be tagged with part-of-speech tags (POS), while others perform POS tagging themselves. Relationship extraction depends on results from sentence boundary detection, gene name recognition, part of speech recognition, and possibly constituent tree parsing and dependency parsing. Also, simple steps like abbreviation detection depend on preprocessing steps like sentence detection. On the other hand, tools also create different types of output which all need to be parsed and transformed into a uniform representation. For instance, some NER tools create inline annotations, i.e., they output a new version of the input text with tags assigned to tokens, while others only create lists of detected entities with references into the text. These references may count tokens or characters; and may refer to different tokenizations and different treatment of special characters, which often requires a complicated re-mapping of detected entities. Upon building the system, such problems take much more time than the simple wrapping of code into an IE pipeline.

Another problem in the application of text mining tools to large collections is their instability in terms of achieved performance. NER (and RE) tools typically are evaluated on small gold standard corpora (GSC) only, which are also used to train the systems. Accordingly, the obtained measures are only valid for these GSC. However, if a GSC has properties deviating substantially from the texts a tool is applied to, very different accuracies may be observed [VCL+12]. When building a system like GeneView which annotates millions of texts, one immediately runs into this problem when inspecting some of the results. For instance, RE algorithms often are developed with GSC that contain a substantially higher fraction of true relationships than ordinary texts; this creates a tendency in classification-based methods to overestimate the a-priori probability of observing a relationship when judging an entity pair, which in turn leads to many false positives [CJK04]. We experimented with simply increasing the confidence threshold for PPI to reduce this problem, but yet did not find a satisfying solution.

In NER, this problem appears in two flavors. First, GSC often contain sets of sentences stemming from different abstracts. Second, most GSC draw their sentences only from abstracts and not from full text. As a consequence, effects of abbreviations are not properly represented (abbreviations are usually defined only once in a text and then used consistently), and the "one-sense-per-discourse" rule is not implemented in NER tools (meaning that a given, generally ambiguous, name usually is used in only one of its senses in a given text). We counteract this effect by two measures. First, when a NER tool tags a long (short) form of an abbreviation and we have detected the abbreviation itself, we also tag the respective short (long) form. Notably, this simple method adds 2,1 million additional gene terms. Second, when a NER tool tags a given token (or set of tokens) and we detect this token again in the same text, we also tag it. The effect of this trick is even more pronounced, as it adds 16,7 million additional gene annotations. These two post-processing steps together are responsible for 50.7% of all visualized gene mentions and have an enormous effect on the user-perceived recall and subsequent relationship extraction – yet a negligible effect when applied to an evaluation on GSC. However, the propagation again is not as simple as it appears, as one has to carefully decide when a subsequent match in a text is "good enough" for receiving an annotation. This is non-trivial, as, on one hand, names for the same gene may differ slightly (e.g. ABC-2 and ABC2 (Entrez Id 20) or TGD and TgD (Entrez Id 19)), while, on the other hand, slight variations in gene names may be decisive (e.g. Fas (Entrez Id 355) and Fas-L (Entrez Id 356)).

Unique concept identifiers, like Entrez gene or dbSNP, are useful to disambiguate entities. However, a user often does not know the identifier associated with the concept she is looking for. We therefore generate a lookup table consisting of all recognized entities including detected synonyms inside of the relational database for all entity types.

For user queries, GeneView proposes the most likely concept by performing a database query and sorting results by the number of articles the entity is found in.

Another problem of large-scale text mining is that some errors are only observed on a small subset of articles, which makes detecting them very hard. Examples are the following. (1) Our abbreviation detection algorithm has problems with different character encodings within the same article, a situation occurring extremely infrequently in PubMed. (2) Some of the NER tools occasionally tag trailing spaces, leading to inconsistencies in visualization. The XML format of PubMed is continuously modified, leading to unexpected parser break-downs (which are spotted immediately) or scrambled visualization (which we cannot detect automatically). (3) For full texts, we keep the XML provided by the publishers to support a journal-specific visualization, leading to diversity in, for instance, the way formulas are represented: Some journals integrate formulas as figures, whereas others enforce the use of MathML, which is removed by our parser in the cleansing step. (4) For dependency parsing, we apply the Charniak Lease parser [LC05] using the McClowsky reranking model [McCH06] which is unable to parse 14,618 out of the total number of 8,131,441 sentences. The reasons for its problems are not clear, yet; it is, however, noteworthy that the large majority (14,546) of problematic sentences came from full-text articles, although the majority of sentences are from abstracts. Again, the original parser is trained on sentences derived from abstracts, which are known to be different from full-text sentences [CJV+10]. This problem required changes in the source code, as the parser stopped after seeing a problematic sentence and did not continue parsing. (5) Additionally, Charniak Lease parser replaces some tokens in the original text by other tokens, further complicating the mapping from parsed text to original text (for instance, tokens like <“> are replaced by <'>).

Computational Requirements

GeneView is regularly updated using a server with 24 cores at 2.6 GHz and 256 GB main memory. Time intensive tasks, especially XML parsing, NER, syntactic parsing, and PPI extraction, are performed in parallel on chunks of the corpus. The computational requirements it takes to rebuild GeneView on a single core are shown in Table 1. Overall, running the entire pipeline in this mode would require an estimated time of 120 days. The by far most time intensive task is syntactic and dependency parsing, although we actually only parse those sentences which mention at least two genes. Of all our NER tools, gene NER is the most time intensive due to its sophisticated disambiguation strategy responsible for mapping a gene mention to its correct database identifier (especially to the correct species). Overall disc space requirement is about 77GB for the Lucene index and 63GB for the metadata and result database.

Processing step	Time [Min]	Size [MB]
Text indexing	1,211	77,855
HTML conversion	528	24,576
Sentence detection	280	9,869
Gene NER	24,012	5,266
SNP NER	14,745	1,986
Histone modification NER	8,090	1,437
AliBaba dictionaries	1,380	5,946
Chemical NER	1,272	16,539
Species NER	1,170	4,927
Abbreviations	727	2,986
Parsing	100,437	44,521
RE detection	11,520	29,483
DB import	3,858	-
Lookup information	1,849	53
Enrich Lucene index	453	-

Table 1. Time (single core) and space requirements to create the complete GeneView index.

4. Indexing Text and Entities

GeneView uses different technologies to store and index its content and to process queries: Lucene is used as a keyword search index and ranking engine; a relational database stores the structured annotation produced by the information extraction pipeline; and a web application interfaces the stored content to the user using the Catalyst MVC framework[5]. While much of the functionality required for GeneView is provided off-the-shelve by the underlying system (for instance, ranking by publication date, PMID, or classical document relevance given a keyword query are directly supported by Lucene), some features of our system require special attention. These are discussed in the following.

Document indexing and ranking

Ranking and filtering functions generally are implemented using Lucene. However, Lucene in the first place is not aware of the counts of detected entities and relationships within a document. Furthermore, ranking by entity-content is not a native feature of Lucene. To achieve this functionality, aggregated text-mining results for each article have to be propagated into the Lucene index and represented properly to integrate them into the customizable ranking mechanism. This encompasses the number of recognized

[5] http://www.catalystframework.org

distinct entities for each type as well as identifiers of recognized entities for each article section. The number of distinct entities of a specific type is used to filter articles without any entity of interest and to rank results by the number of distinct entities. The information about identifiers found in a specific article enables users to search for articles containing specifically this entity of interest (regardless of homonyms, synonyms etc.).

For gene queries, the query relevance ranking is modified and a section specific ranking is applied. Optimal section weights have been determined using PubMed's gene2pubmed as described in [TSJ+10]. Gene2pubmed provides manually curated links between PubMed articles and the genes contained in them. Using this data, we set weights as Lucene boost parameters such that a query for a curated gene in gene2pubmed ranks the corresponding articles in gene2pubmed highest. This strategy allows us to estimate the average ranking quality for gene queries and considerably improved the mean average precision of gene queries. The automatically derived section weights meet general expectations in that, for instance, sections like *Title* are highly ranked, while *Materials and Methods* receive low weights. Technically, it would also be possible to extend this functionality to other types of entities; however, we currently see no sensible method to obtain rational weights for entities other than genes. Furthermore, the corresponding boosts would either interfere with each others, possibly blurring the intended improvements, or be provided separately at the user interface, which again would make it more complicated.

To allow users to focus on their particular set of genes, GeneView allows the definition of individual gene lists which later can be used to filter/rank articles of any query. In such cases, the query is expanded with the members of the gene set; implementing this feature therefore only requires functionality for storing and managing personalized gene lists, while their integration into the ranking can be achieved with standard Lucene methods. Note that achieving this functionality manually would be hard, as such gene lists often contain dozens or even hundreds of genes (in case of genetically complex diseases such as cancer or diabetes). It would be conceptually straight-forward to expand this feature to types of entities other than genes, but therein one carefully has to balance functionality and simplicity of the user interface. Currently, most users we talked to have a strong focus on genes.

Another feature of GeneView important for users is "rank by entity count". To this end, we extract aggregated counts from the database and store them as additional metadata in a proper Lucene field attached to each document. At query time, one can tell Lucene to use the information in this field for ranking and/or filtering. This solution works equally well for all types of counts; however, for usability reasons we currently expose this functionality only for SNPs and genes at the web interface.

Annotation indexing

All entities and relationships extracted by the extraction pipeline are stored in a relational database. Information stored for each entity mention includes the article id, normalized entity id, concrete annotated text span, start character position in the cleansed text, and end character position in the cleansed text. The article Id, which is the PubMed article identifier (PMID), links each mention to the corresponding document in the Lucene index. The normalized entity Id links a mention to additional information in external, type-specific data sources (e.g., Entrez-Gene-Id for genes or Chebi-Id for chemicals). The annotated text span and the start and end positions precisely define the actual occurrence of the entity in the inspected document. This information is used for entity highlighting when visualizing single articles, which requires an additional step of mapping character positions as stored in the database to the HTML representation of the text created from the original XML files. Due to the multiple text manipulations that take place in-between, those mappings cannot be computed automatically; instead, we have to retain a positional mapping table for each inspected document (see Section 3). For all relationship types, we store links to the two linked entities, classifier confidence, and associated sentence. Note, that the resulting database is quite large; altogether, GeneView indexes more than 209 million different entity mentions.

Entity type	Entities	Distinct entities	Number of articles
Cell-type	18,891	231	5,622
Chemical	77,606,023	47,905	9,851,536
Disease	145,001	4,643	74,583
Drugs	47,113,224	3,061	6,246,067
Enzyme	894,895	2,298	590,301
Genes	37,080,749	83,705	2,959,439
Histone-mod	77,210	575	7,673
SNP	1,078,640	42,505	192,544
Species	44,808,988	115,966	9,119,134
Tissue	239	31	222
Overall	209,788,411	304,565	13,463,850

Table 2. Overview of detected entities in GeneView.

Document specific aggregated information for each entity type is injected into the Lucene index once the NER/NEN/RE pipeline has finished. Thereby, Lucene can handle all ranking issues without a need to get back to the databases; the database is only accessed for highlighting during web display (see above) and for assisting users in formulating queries. Here, GeneView provides on-the-fly lookup functionality which suggests auto-completions if entered tokens match an entity name (see also Section 2). This lookup issues one query to the database for each keystroke the user makes, which in turn requires a carefully indexed lookup table. We realize this lookup as a materialized

view over the entity-specific annotation tables storing the original mention, its normalized representation and its corresponding identifier. Additionally, each entry contains the overall number of occurrences of this entity in the corpus and the number of articles it has been found in.

Visualization in the Web Interface

For single article visualization, entities and their spans are requested from the relational database. For each type of entity found, a separate instance of the articles HTML representation is enriched with highlighting in a type-specific color. When displayed in the browser, these instances are overlaid to appear as a single document. The objective of this multi-layered approach is to allow collision free multi-entity annotation. For instance, a single entity may be (correctly) identified as both a drug and a chemical, causing two overlapping annotations. As GeneViews highlighting are semi-transparent, the resulting overlap of layers will appear to the user in a different, mixed color, indicating the detected ambiguity. A drawback is the need to transfer each text to the user, i.e., from server to client, multiple times within a single HTML document. While this is less problematic for abstracts, it does raise scalability issues for lengthy full texts in terms of the number of different entity types which can be included. For instance, GeneViews web page of a full text including five different types of entity mentions can reach a size of around 1MB.

For its aggregate view of annotations in the left-hand pane, GeneView utilizes the entity-count data stored in the Lucene index. For each entity type, it shows a table of all found entities ranked by their number of occurrences. For genes, these counts are also available on the section level. This enables the user to fine-tune the ranking by including or excluding specific sections.

The web interface also provides link-out information for entities. Such information is implicitly linked to an entity during entity normalization; the source of the specific information is type-dependent. For instance, information on genes is retrieved from the PiPa database [ASA11]. This information is aggregated and enriched with external links to the referenced databases. The specific information to be displayed is configured by implementing a plug-in for the respective entity type. When an entity is clicked in the user interface, an AJAX call with entity type and entity Id is sent to the web server, which uses this information to resolve the plugin to be used, which in turn retrieves the information to be displayed and converts it into a HTML snippet sent back to the client. Generally, such plug-ins can be arbitrarily complex. While some simply generate links to external reference databases, others may retrieve information from such databases at run-time, such as for genes (see above).

5. Conclusions

We presented GeneView, an entity-centric search engine for the biomedical literature. To achieve its functionality, the system encompasses over two dozens of external NLP and information extraction tools whose output are stored in a classical information retrieval engine (Lucene) and in a relational database (MySQL).

This paper gave an account of the many smaller and larger problems that emerge during the construction of systems like GeneView. Many of these problems stem from the fact that we follow a best-of-breed strategy, i.e., we use the best available tools for each of the different entity classes and relationship types that are indexed, which comes along with heterogeneous requirements in terms of execution environment, different data formats, multiple runtime dependencies, and continuous problems with version incompatibilities. In particular, the lack of standards for representing annotated texts, which gives rise to many different ways to link annotations with text spans, creates the need to perform repeated format conversions and to keep multiple copies of the text, along with brute-force mapping tables. Almost every tool in our pipeline has a different format for the input text and the positional annotations it returns. We currently see little hope that these problems will go away in the near future, unless efforts such as ???NIF succeed in defining standards for the community. As a positive message, we experienced that the basic infrastructures, especially Lucene, are able to provide stable, flexible and scalable search performance, although their usage for advanced features such as entity-based ranking requires some thought and effort.

However, we also see that a project like GeneView poses considerable challenges to current methods in terms of scalability, flexibility, and maintenance cost. For instance, the workflow depicted in Figure 5 can be executed in various orders, each of which will take different time depending on the selectivity of the contained filter operations, the time required to execute the various tools on input of varying size, the available hardware, etc. There have been first attempts to optimize such complex IE workflows mostly consisting of non-standard operations [RRK+08; SDNR07], but these focus on comparably simple operations like regular expression matching and co-occurrence. We believe that advanced methods for entity recognition and relationship extraction like the one implemented in GeneView have distinct properties that call for specific optimization techniques. We have started work in this direction [HRL+12] in the course of the Stratosphere project[6].

Another challenge is flexibility in executing an IE pipeline. Very often, only parts of the entire workflow have to be run, for instance if new versions of individual tools are available. In such cases, running the entire workflow would imply a great deal of

[6]See http://www.stratosphere.eu/

unnecessary computations, but running only specific parts of it is not easily achieved, given that the workflow technically consists of a series of intertwined scripts in different languages. But because implementing sub-workflows is costly in terms of manpower, we often simple run the entire workflow despite the waste in compute power. A proper support for specifying and executing such pipelines should also support data incremental execution, as pipelines often break unexpectedly due to format problems in the input or bugs in the IE tools. Restarting the pipeline should not imply re-annotating texts that had already been finished in the previously though finally failed run. There exist some suggestions towards this problem [KSB+10], but these, to the best of our knowledge, haven't been integrated into real dataflow languages yet.

Besides these data-management-related challenges, also the text mining part of GeneView is still far from fully satisfactory, especially in the area of relationship detection. For instance, relationship extraction still is sentence-specific. Finding relationships across sentences would require the inclusion of algorithms for anaphora resolution [HK09]. Another problem is that of negation and hedging, i.e., assessing the strength of the author's certainty in a reported, possibly negative, finding [FVM+10].

Acknowledgements

We thank Astrid Rheinländer, Sebastian Arzt, Mariana Neves, Alexander Vowinkel, and Tim Rocktäschel for contributions to GeneView. This work was partly funded by the DFG, grants GRK1651 (SOAMED) and LE 1428/4-1 (Stratosphere), and by the BMBF, grant 0315417B (ColoNet).

References

[APB+08] Airola, A., Pyysalo, S., Bjorne, J., Pahikkala, T., Ginter, F. and Salakoski, T. (2008). "All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning." BMC Bioinformatics 9 Suppl 11: S2.

[ASA11] Arzt, S., Starlinger, J., Arnold, O., Kröger, S., Jaeger, S. and Leser, U. (2011). "PiPa: Custom Integration of Protein Interactions and Pathways". Workshop Daten In den Lebenswissenschaften, Berlin, Germany.

[Bal12] Balke, W.-T. (2012). "Introduction to Information Extraction: Basic Notions and Current Trends." Datenbank-Spektrum 12(2).

[BKS10] Bui, Q. C., Katrenko, S. and Sloot, P. M. (2010). "A hybrid approach to extract protein-protein interactions." Bioinformatics.

[CBR+07] Caporaso, J. G., Baumgartner, W. A., Randolph, D. A., Cohen, K. B. and Hunter, L. (2007). "MutationFinder: a high-performance system for extracting point mutation mentions from text." Bioinformatics 23(14): 1862-1865.

[CJK04] Chawla, N. V., Japkowicz, N. and Kotcz, A. (2004). "Editorial: special issue on learning from imbalanced data sets." ACM SIGKDD Explorations Newsletter 6(1): 1-6.

[CL10] Chowdhury, F. M. and Lavelli, A. (2010). "Disease Mention Recognition with Specific Features". Workshop on Biomedical Natural Language Processing, Uppsala, Sweden.

[CJV+10] Cohen, K. B., Johnson, H. L., Verspoor, K., Roeder, C. and Hunter, L. E. (2010). "The structural and content aspects of abstracts versus bodies of full text journal articles are different." BMC Bioinformatics 11: 492.

[DWH10] Divoli, A., Wooldridge, M. A. and Hearst, M. A. (2010). "Full text and figure display improves bioscience literature search." PLoS One 5(4): e9619.

[DMNL09] Dogan, R. I., Murray, G. C., Névéol, A. and Lu, Z. (2009). "Understanding PubMed® user search behavior through log analysis." Database (Oxford).

[FVM+10] Farkas, R., Vincze, V., MÃ³ra, G., Csirik, J. and Szarvas, G. (2010). "The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text". Conf. on Computational Natural Language Learning - Shared Task Stroudsburg, US. pp 1-12.

[GNB10] Gerner, M., Nenadic, G. and Bergman, C. M. (2010). "LINNAEUS: a species name identification system for biomedical literature." BMC Bioinformatics 11: 85.

[GSBN12] Gerner, M., Sarafraz, F., Bergman, C. M. and Nenadic, G. (2012). "BioContext: an integrated text mining system for large-scale extraction and contextualisation of biomolecular events." Bioinformatics 28(16): 2154-2161.

[GLR06] Giuliano, C., Lavelli, A. and Romano, L. (2006). "Exploiting shallow linguistic information for relation extraction from biomedical literature". European Chapter of the Association for Computational Linguistics Trento, Italy. pp 401–408.

[HK09] Haghighi, A. and Klein, D. (2009). "Simple coreference resolution with rich syntactic and semantic features". Int. Conf. on Empirical Methods in Natural Language Processing, Singapore. pp 1152-1161.

[HGH+11] Hakenberg, J., Gerner, M., Haeussler, M., Solt, I., Plake, C., Schroeder, M., Gonzalez, G., Nenadic, G. and Bergman, C. M. (2011). "The GNAT library for local and remote gene mention normalization." Bioinformatics 27(19): 2769-2771.

[HRL05] Hakenberg, J., Rutsch, J. and Leser, U. (2005). "Tuning text classification for hereditary diseases with section weighting". Symposium on Semantic Mining in Biomedicine (SMBM), Hinxton, UK. pp 34-39.

[HRL+12] Heise, A., Rheinländer, A., Leicht, M., Leser, U. and Naumann, F. (2012). "Meteor/Sopremo: An Extensible Query Language and Operator Model". Workshop on End-to-end Management of Big Data, Istanbul, Turkey.

[KSB+10] Koop, D., Santos, E., Bauer, B., Troyer, M., Freire, J. and Silva, C., T. (2010). "Bridging Workflow and Data Provenance Using Strong Link". Int. Conf. on Scientific and Statistical Database Management Systems, Heidelberg, Germany.

[KMS+08] Krallinger, M., Morgan, A., Smith, L., Leitner, F., Tanabe, L., Wilbur, J., Hirschman, L. and Valencia, A. (2008). "Evaluation of text-mining systems for biology: overview of the Second BioCreative community challenge." Genome Biol 9 Suppl 2: S1.

[LC05] Lease, M. and Charniak, E. (2005). "Parsing Biomedical Literature". Second International Joint Conference on Natural Language Processing (IJCNLP'05).

[LH05] Leser, U. and Hakenberg, J. (2005). "What Makes a Gene Name? Named Entity Recognition in the Biomedical Literature." Briefings in Bioinformatics 6(4): 357-369.

[Lu11] Lu, Z. (2011). "PubMed and beyond: a survey of web tools for searching biomedical literature." Database (Oxford) 2011: baq036.

[McCH06] McClosky, D., Charniak, E. and Johnson, M. (2006). "Reranking and self-training for parser adaptation". Int. Conf. on Computational Linguistics, Stroudsburg, USA. pp 337-344.

[OW04] Ogino, S. and Wilson, R. B. (2004). "Importance of standard nomenclature for SMN1 small intragenic ("subtle") mutations." Human Mutation 23(4): 392-393.

[RRK+08] Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H. and Vaithyanathan, S. (2008). "An Algebraic Approach to Rule-Based Information Extraction". 24th International Conference on Data Engineering, Cancun, Mexico. pp 933-942.

[RWL12] Rocktäschel, T., Weidlich, M. and Leser, U. (2012). "ChemSpot: A Hybrid System for Chemical Named Entity Recognition." Bioinformatics 28(12): 1633-1640.

[Sar08] Sarawagi, S. (2008). "Information Extraction." Foundations and Trends in Databases 1(3): 261-377.

[SH03] Schwartz, A. S. and Hearst, M. A. (2003). "A simple algorithm for identifying abbreviation definitions in biomedical text". Pacific Symposium on Biocomputing, Hawaii, US.

[SDNR07] Shen, W., Doan, A., Naughton, J. F. and Ramakrishnan, R. (2007). "Declarative Information Extraction Using Datalog with Embedded Extraction Predicates". Int Conf. on Very Large Databases, Vienna, Austria. pp 1033-1044.

[TNS+11] Thomas, P., Neves, M. L., Solt, I., Tikk, D. and Leser, U. (2011). "Relation Extraction for Drug-Drug Interactions using Ensemble Learning". DDIExtraction Workshop, Spain.

[TSJ+10] Thomas, P., Starlinger, J., Jacob, C., Solt, I., Hakenberg, J. and Leser, U. (2010). "GeneView: Gene-Centric Ranking of Biomedical Text". Proceedings of BioCreative III, Bethesda, USA. pp 137-142.

[TSV+12] Thomas, P., Starlinger, J., Vowinkel, A., Arzt, S. and Leser, U. (2012). "GeneView: A comprehensive semantic search engine for PubMed." Nucleic Acids Res 40(Web Server issue): 585-591.

[TTP+10] Tikk, D., Thomas, P., Palaga, P., Hakenberg, J. and Leser, U. (2010). "A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature." PLOS Computational Biology 6(7).

[VCL+12] Verspoor, K. M., Cohen, K. B., Lanfranchi, A., Warner, C., Johnson, H. L., Roeder, C., Choi, J. D., Funk, C., Malenkiy, Y., Eckert, M., et al. (2012). "A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools." BMC Bioinformatics 13(1): 207.

[ZDF+08] Zweigenbaum, P., Demner-Fushman, D., Yu, H. and Cohen, K. B. (2007). "Frontiers of biomedical text mining: current progress." Brief Bioinform 8(5): 358-75.