

Potential and Pitfalls of Domain-Specific Information Extraction at Web Scale

Astrid Rheinländer*, Mario Lehmann, Anja Kunkel, Jörg Meier, and Ulf Leser*
Humboldt-Universität zu Berlin, Department of Computer Science, Berlin, Germany
*{rheinlae,leser}@informatik.hu-berlin.de

ABSTRACT

In many domains, a plethora of textual information is available on the web as news reports, blog posts, community portals, etc. Information extraction (IE) is the default technique to turn unstructured text into structured fact databases, but systematically applying IE techniques to web input requires highly complex systems, starting from focused crawlers over quality assurance methods to cope with the HTML input to long pipelines of natural language processing and IE algorithms. Although a number of tools for each of these steps exists, their seamless, flexible, and scalable combination into a web scale end-to-end text analytics system still is a true challenge.

In this paper, we report our experiences from building such a system for comparing the “web view” on health related topics with that derived from a controlled scientific corpus, i.e., Medline. The system combines a focused crawler, applying shallow text analysis and classification to maintain focus, with a sophisticated text analytic engine inside the Big Data processing system Stratosphere. We describe a practical approach to seed generation which led us crawl a corpus of ~1 TB web pages highly enriched for the biomedical domain. Pages were run through a complex pipeline of best-of-breed tools for a multitude of necessary tasks, such as HTML repair, boilerplate detection, sentence detection, linguistic annotation, parsing, and eventually named entity recognition for several types of entities. Results are compared with those from running the same pipeline (without the web-related tasks) on a corpus of 24 million scientific abstracts and a third corpus made of ~250K scientific full texts. We evaluate scalability, quality, and robustness of the employed methods and tools. The focus of this paper is to provide a large, real-life use case to inspire future research into robust, easy-to-use, and scalable methods for domain-specific IE at web scale.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-3531-7/16/06...\$15.00

DOI: <http://dx.doi.org/10.1145/2882903.2903736>

Keywords

Focused crawling; Information extraction; Massively parallel data analysis

1. INTRODUCTION

The analysis of information published on the web is valuable for a plethora of applications, e.g., to analyze customer product reviews [21], to investigate relationships between politicians and their sponsors [12], or to predict flu waves and assess their treatments [7], to name just a few. Analyzing web data is not trivial due to its scale, distribution, heterogeneity, redundancy, and questionable quality. Compared to traditional text analytics, already obtaining the data to be analyzed is difficult, requiring either access to an existing (open) large web crawl or the set-up and running of a proper crawler. For applications requiring domain-specific texts, like the one we focus on here, special care must be taken to restrict the crawl to this domain, typically by applying text classification on the crawl or during the crawling [8]. Another severe issue arises from the extreme heterogeneity of web documents and their cluttering with navigational elements, advertisements, script code, formatting instructions etc. [29]. Many projects circumvent this problem by focusing on a single or a few data sources, typically the big social media platforms such as Twitter, Facebook or Flickr; however, this excludes literally billions of additional knowledge sources. Finally, the filtered and cleansed web texts must be analyzed by information extraction (IE) algorithms to obtain the desired facts, which in itself is a challenging task when the text collection is large and the requirements regarding data quality are high.

In this paper, we report on our experiences from building a comprehensive system for domain-specific text analytics on the open web. Building such systems for long was only possible for large web companies; however, advances in cloud computing, IE, and crawler techniques together with falling prices for storage, computing power, and network bandwidth put such systems – in principle – also into the realm of mid-size organizations. But putting this theoretical possibility into practise still is a highly challenging task. Therefore, the goal of this experience paper not only is to describe such a system, with a focus on design issues regarding robustness, data quality, and scalability, but also to pinpoint the most critical issues that need to be solved when building such systems, with the ultimate intention to foster more research into this important and challenging area.

As a concrete use case, we applied the system to crawl a 1 TB collection of web text from the biomedical domain

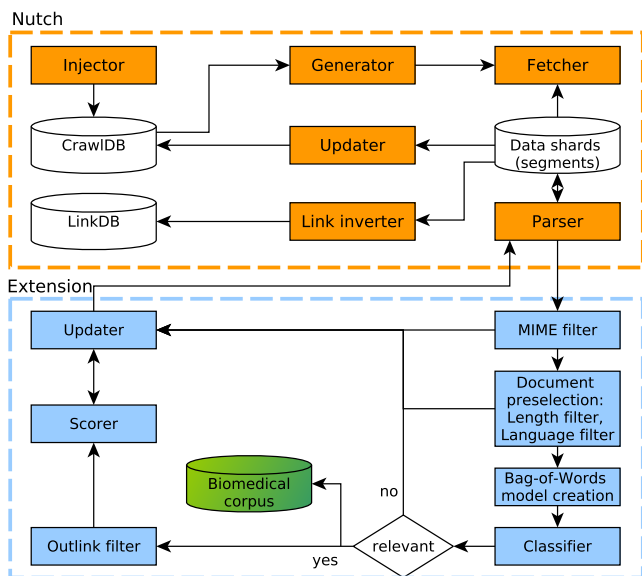


Figure 1: Architecture of a topical crawler based on Apache Nutch.

with the goal to retrieve a high quality corpus in terms of precision with respect to our target domain. This corpus was cleansed and filtered by specific modules for web texts, linguistically preprocessed using methods from statistical natural language processing (NLP), and eventually analyzed by a series of domain-specific IE programs to find mentions of important entities in this field, such as genes, drugs, or diseases. We then ran the same pipeline on two much more controlled sets, i.e., all abstracts in the Medline collection of scientific articles, and a set of approximately 250.000 biomedical full texts. A fourth corpus was built from all web pages deemed out-of-domain by the focused crawler. Next, we compared results from a linguistic analysis and from the domain-specific IE on the four corpora to each other, finding notable differences in many aspects, including simple metrics such as average sentence and document length, more linguistically motivated properties such as the use of negation or abbreviations, and, eventually, the sets and frequencies of occurring domain-specific entities. The system applies advanced machine learning (ML) in every phase of its collection and analysis pipeline, i.e. text classification during the focused crawling, snippet classification for the extraction of net text from HTML pages, sequential classification with Hidden Markov Models for the NLP, and classification, pattern matching and Conditional Random Fields for the IE tasks. Most notably, the entire process for web text analysis (excluding the crawling) was specified, optimized, and executed using a small set of data flows in a single, homogeneous, and declarative framework for specifying UDF-heavy data flows, i.e., the Stratosphere system [2, 23]. Using this framework allowed us to systematically evaluate the entire extraction process with respect to scalability, efficiency, and quality of the involved tools. We believe our approach implements a notable advancement compared to the current state-of-the-art for building such systems, which boils down to manually created scripts implementing an ad-hoc assembly of existing tools. This practice clearly interferes with today’s needs in Big Data

analytics; instead, we envision complex information acquisition and extraction from the web as an almost effortless end-to-end task.

The remainder of this paper is structured as follows: Section 2 explains the retrieval of biomedical web pages by means of focused web crawling at large scale. Section 3 reviews the data analytics platform, discusses data flows we used for analyzing the crawl. Section 4 evaluates efficiency and scalability of crawling and presents results of our analysis with respect to language structure and contents of the crawled documents. Section 5 summarizes lessons learned in this study and highlights open engineering and research challenges for efficient text analytics at large scale. We conclude this paper in Section 6.

2. FOCUSED CRAWLING: THE BASIS FOR DOMAIN-SPECIFIC TEXT ANALYTICS

The goal of our research is to perform advanced IE on domain-specific collections of web documents. A proper way to obtain such a collection are focused crawlers [8]. A focused web crawler traverses parts of the web to find documents relevant for a certain topic. To speed-up the crawling and to obtain good harvest rates (i.e. a large density of relevant pages among all crawled pages), a major objective during focused crawling is to visit only those outgoing links of a website that appear to be particularly relevant for a given topic. To decide whether a link is relevant or not, the assumption is made that relevant pages are most likely linked to other relevant pages whereas irrelevant pages point more often to other irrelevant pages and thus constitute an endpoint during the crawl. To assess the relevance of page, a focused crawler is equipped with a text classifier trained on a set of pre-classified documents. We built a focused crawler which follows this approach: It downloads web pages, classifies them as relevant or not, and only further considers links outgoing from relevant pages. We did not follow the alternative approach of classifying links based on its surroundings because this would require the laborious creation of a training corpus of links; in contrast, obtaining a training corpus of relevant documents is comparably simple. For our study, we trained on a set of randomly selected abstracts from Medline¹), considered as relevant, and an equal-sized set of randomly selected English documents taken from the common crawl corpus²), considered as irrelevant. This approach is cheap and simple; note, however, that it introduces some bias as a typical Medline abstract is quite different from a typical web page (see Section 4.3.1).

2.1 Crawler architecture

To obtain a large corpus of reasonable quality, the setup of the focused crawler is crucial. Web crawlers should be unsusceptible to so-called spider traps, i.e., websites containing programming errors or dynamically generated links that cause the crawler to be trapped in an infinite loop. A crawler also needs to respect the implicit and explicit rules of a domain (e.g., maximum number of simultaneous requests, rules contained in the “robots.txt” file). Finally, it must be implemented in a distributed manner to allow for using multiple machines in parallel. There exist a number of frameworks which implement such functionality; we built our system on

¹<http://www.ncbi.nlm.nih.gov/pubmed>

²<http://www.commoncrawl.org>

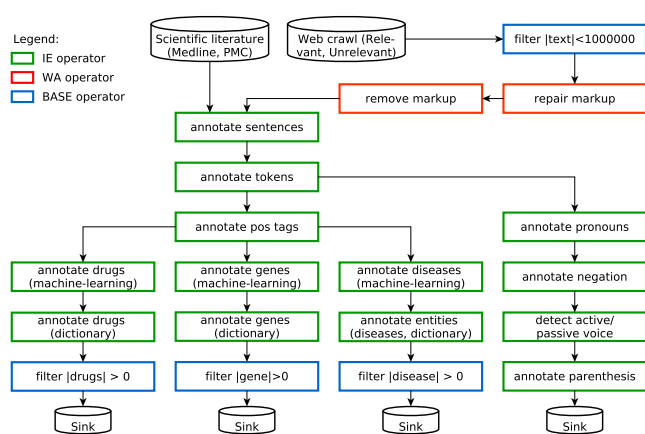


Figure 2: Consolidated high-level data flow for analyzing biomedical documents from scientific publications and the internet. Isochromatic frames indicate Sopro operator package.

top of the open-source framework Apache Nutch³. Nutch is based on Apache Hadoop⁴ to enable scalable and distributed crawling. It lacks a component for focusing a crawl, but has a clean extension interface which we used to plug-in a classifier and the necessary logic. Figure 1 shows the architecture of Nutch together with our custom extensions. The part implemented in Nutch (upper part of Fig. 1) is fairly conventional; an injector reads seed URLs from a text file and adds these to the crawl database (CrawlDB) acting as the crawl frontier. A set of fetcher threads reads lists of not yet visited URLs from this database, downloads the respective web pages and forwards them to a parser for link and content extraction; unseen links are added to the CrawlDB. Besides, a link database (LinkDB) stores the graph structure of the crawled pages.

To add focus to the crawling process, we extended Nutch with the following components (lower part of Fig. 1): After parsing a web page, we first check whether it is of textual content using a MIME type filter. We also remove pages that are too short and pages that are written in languages other than English by using an n-gram based language filter, because subsequent IE tools used in this analysis are sensitive to language. Afterwards, the main text of a page is extracted using the tool Boilerpipe [15]. To classify a document, its extracted net text is converted to a Bag-of-Words model and classified for its biomedical relevance. We use a Naïve Bayes algorithm due to its robustness with respect to class imbalance (we have no rational guess on the expected percentage of biomedical pages during a focused crawl) and its ability to update its model incrementally, although we currently don't use this feature. If a page is classified as relevant, it is added to the corpus and all its outlinks are added to CrawlDB; otherwise, it is discarded. The crawling and classification process is repeated iteratively until either CrawlDB is empty, the desired corpus size is reached, or it is stopped manually by the user.

2.2 Seed generation

A very important issue in crawling, and especially in fo-

³<http://nutch.apache.org>

⁴<http://hadoop.apache.org>

cused crawling, is the determination of the set of seed URLs used to initiate the crawl. The typical way of obtaining a large set of seeds is to issue keyword queries to one or more search engines. For focused crawling, keywords are chosen such that they retrieve domain-specific seeds with high probability. Since all search engine APIs restrict the number of allowed queries and limit the number of returned results, one often uses (a) multiple search engines in parallel and (b) large sets of queries - which creates the necessity to generate thousands of high quality queries. For our case study, we utilized five different search engines, namely Bing, Google, Arxiv, Nature, and Nature blogs. For each search engine, we generated queries with (a) general biomedical terms, obtained from National Cancer Institute⁵ and the Genetic Alliance glossary⁶ and (b) highly specific molecular terms extracted from the Gene Ontology⁷, Drugbank⁸, and the UMLS/MeSH sub-tree for diseases⁹. Exemplary keywords are shown in Table 1 together with the total number of search terms for each category. Clearly, the chosen queries give the resulting corpus a certain direction; in our case, we intended to focus on genetic facts about diseases.

In a first experiment, we used only a subset of keywords from our data sources (see Table 1, numbers in bracket). All search results from the different search engines obtained with these keywords were merged to a single list of 45,227 seed URLs. However, the resulting crawl terminated quickly due to an emptied CrawlDB, i.e., all pages in the frontier of pages reachable from these seeds were classified as irrelevant. Debugging this crawl, we found several reasons for this situation. First, the search terms chosen were too general. For these, the search engines return rather general pages, which they considered as authoritative for the respective topic, such as front pages of portals. These pages very often immediately were classified as irrelevant, i.e., this branch of the crawl stopped after just one step. Second, we found that biomedical sites generally are only weakly linked; most often, all outgoing links from a page were navigational leading to pages on the same host (see Sect. 4.1 for details).

Note that not stopping the crawl of such irrelevant pages immediately but after n steps (e.g., $n = 2$, $n = 3$) is a viable alternative to increase the size of the crawl, since many front pages of portals deemed irrelevant link to relevant content for our domain. Yet, crawling time will significantly increase since many irrelevant pages will be explored as well. Since the entire crawling process already stretched to more than two months while stopping immediately when visiting an irrelevant page (see Section 4.1 for details), we decided to increase the set of seed URLs to obtain a larger crawl instead. Consequently, we performed a second seed generation run, this time using 15,000 queries resulting in a total number of 485,462 seed URLs. These seeds were used for crawl described previously.

3. DATA FLOWS FOR WEB-SCALE INFORMATION EXTRACTION

Information extraction (IE) and statistical natural language processing (NLP) tasks on large text collections are

⁵<http://www.cancer.gov>

⁶<http://www.geneticalliance.org.uk/glossary.htm>

⁷<http://geneontology.org>

⁸<http://www.drugbank.ca>

⁹<http://www.nlm.nih.gov/mesh/>

Category	No. of terms	Example search terms
general terms	500 (166)	cancer, chronic pain
disease-specific	5000 (468)	thymoma, nausea, cough
drug-specific	4000 (325)	GAD-67, Aspirin
gene-specific	6500 (246)	BRCA, Cactin

Table 1: Total number and exemplary search terms by category used for seed URL retrieval. Numbers in brackets denote the number of search terms for the first crawl (see text).

time- and resource-consuming because of the complexity of the involved tasks [5]. Our aim was to apply state-of-the-art techniques in biomedical information extraction. This requires the use of several heavy-weight tools and algorithms, some of which have a runtime complexity that is quadratic in the text length (see below). Due to the embarrassingly parallel nature of information extraction at the entity level, a natural choice to alleviate runtime problems is to parallelize all analysis on a cluster. To this end, we implemented and wrapped a variety of IE and NLP tools for use with Stratosphere¹⁰, a parallel data processing platform for large-scale, UDF-heavy data analytics. A detailed description of Stratosphere can be found in [2], in the following, we very briefly summarize key concepts of Stratosphere and then discuss in detail the data flow we implemented to perform text analytics on the crawled corpus and other texts.

3.1 Stratosphere

Stratosphere is a system for parallel execution of UDF-heavy data flows. These data flows are specified in a declarative scripting language called Meteor (note that there also exist other APIs) [13]. Meteor scripts are composed of primitive operators, which are defined in domain-specific packages, i.e., self-contained libraries of the operator implementations, their syntax, and semantic annotations. A Meteor script is parsed into an algebraic representation, logically optimized [23], and compiled into a parallel data flow program of parallelization primitives (e.g., map, reduce, cross) embracing the operators implementations. Subsequently, the parallel data flow program is physically optimized, translated into an execution graph and deployed on the given hardware [2].

Currently, the system ships more than 60 different operators organized in four packages, i.e., general purpose (BASE), information extraction (IE), web analytics (WA), and data cleansing (DC). The BASE package comprises relational operators, such as filter, projection, transformation, or join. The IE package includes operators for annotating texts with syntactic annotations (e.g., sentence boundaries, part-of-speech tags) and semantic annotations (e.g., mentions of different entity types, relationships between entities), and for merging annotations using different schemes. The WA package contains operators specific to the analysis of web documents, such as link extraction, markup removal, or markup repair. Eventually, the DC package brings operators for data cleansing and data integration, addressing common challenges in processing dirty or heterogeneous data sources.

¹⁰Stratosphere was recently renamed into Flink and is now an Apache Top-Level project; our code builds on release 0.4.2.

3.2 Data flow for Crawl Analysis

The ultimate goal of our study was to compare the domain-specific "information content" of biomedical web pages with that of a corpus of scientific publications, either as full texts or as abstracts. Furthermore, we wanted to compute general linguistic characteristics of the different corpora we studied, such as average sentence length or frequency of negated sentences. The latter has two purposes: First, we find it of general interest to differentiate properties of different types of texts in a given domain. Second, these properties have consequences on the tools which are used to analyze the texts. Eventually, we were also interested in the link topology of the biomedical pages to test some common assumptions on these structures.

Processing a large set of documents to compute a number of measures and to extract a variety of different information requires a complex set of tasks. Specifically, our scenario encompasses three different sub-problems, namely document preprocessing and HTML cleansing, linguistic document analysis, and biomedical entity extraction. Each of these sub-problems requires the use of a bunch of different tools; for instance, high-quality entity extraction is only possible using separate and highly optimized tools for each entity class. Furthermore, the different processing steps have various dependencies. Ideally, all tasks and their dependencies are expressed using a single data flow program to enable a holistic optimization and seamless analysis. Stratosphere allows such a concise specification; Fig. 2 displays the high-level data flow we developed for this case study. The complete data flow comprising all required analysis for this study consists of 38 elementary operators. Web pages are first filtered to exclude extremely long documents. Next, HTML markup is detected, errors are repaired, and all markup is removed from each page. All documents are annotated with sentence and token boundaries. For linguistic analysis, each sentence is analyzed for occurrences of pronouns, negation, and parenthesis using different sets of regular expressions, and each found mention of any of these categories is added to the result set together with information on document ID, sentence ID, and start/end positions. For the biomedical content analysis, we then annotate three types of biomedical entities, namely genes, drugs, and diseases.

Note that biomedical IE is considered particularly challenging because of ambiguous naming conventions, multi-word names, acronyms, etc., and specialized methods, dictionaries, and models are needed to achieve satisfactory results with respect to extraction accuracy [18]. The accuracy of all tools applied to web documents is difficult to predict, since all current tools were trained and evaluated only on scientific articles, and mostly only on abstracts. Therefore, we chose to apply two different extraction methods for each entity type on the entire data set: A classical fuzzy dictionary-matching tool, and ML-based entity taggers using Conditional Random Fields (CRF). In this field, dictionary-based entity extraction typically achieves good precision yet low recall because dictionaries are necessarily incomplete in a field developing as fast as biomedical research. On the other hand, ML-based extraction methods often show much improved recall and they also show superior precision. Besides, they usually exhibit a significantly slower processing speed.

Note that none of the tools we applied was developed for this study; instead, they were chosen from available open

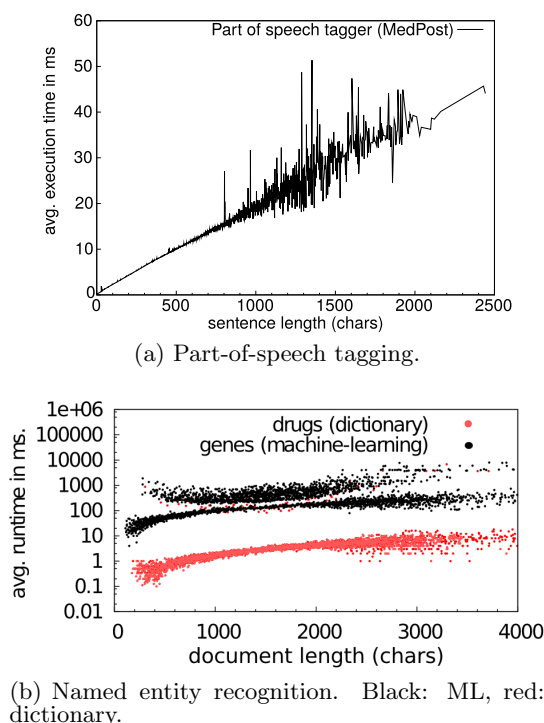


Figure 3: Runtimes information extraction tools with respect to the length of the input texts.

source software following a best-of-breed strategy. For dictionary matching, we used an automaton-based matching algorithm that quickly retrieves mentions of entities even for large dictionaries [11]. To account for some variations, we transformed each dictionary term into a regular expression. The largest dictionary employed here, that for gene names, contains more than 700,000 entries; dictionaries for disease and drug names were significantly smaller with 61,438 and 51,188 entries, respectively. As ML-based tool for drug names we used ChemSpot [24], for gene names we applied BANNER [17], and for diseases we integrated a previously developed tool in our group directly building on the CRF library Mallet¹¹. Note that also ChemSpot and BANNER use Mallet.

4. EVALUATION

In this section, we perform a detailed evaluation of the many facets of our case study in domain-specific web-scale information extraction. Specifically, we first give a technical evaluation of the focused crawl itself, which implies assessing the harvest rate of the crawler, the quality of the text classifier used to keep focus and the quality of the boilerplate detection algorithm we applied. Next, we focus on the performance and scalability of the information extraction pipeline, which includes identifying the most time-consuming steps, contrasting dictionary-based entity recognition with ML-based approaches in terms of recognition speed, and experiments for showing the scale-out our pipeline achieves thanks to the underlying processing system Stratosphere. Eventually, we provide a detailed analysis of our

¹¹<http://mallet.cs.umass.edu>

biomedical web corpus in contrast to three other corpora, i.e., the set of pages classified as irrelevant during the crawl, the complete set of approximately 21 million abstracts from Medline (until year 2013), and a set of approximately 250,000 full texts from the PLoS open data mining collection (PMC).

4.1 Quality of the Focused Crawler

The premise of our study is to enable seamless information extraction on large sets of crawled, domain-specific documents on a mid-size cluster. We run our crawler on a cluster of 5 servers, each equipped with at least 32 CPU cores and connected with a 1 GB line to a 10 GB switch connected to the Internet backbone. Politeness rules of web servers were respected and the sizes of host-specific fetch lists was limited to 500 to prevent threads from blocking each other. Each downloaded page passing the initial filtering was subjected to boilerplate removal and to text classification. With this set-up, our crawler achieved a download rate of 3-4 documents per second, which is notably slower compared to other systems (e.g., [20] considers download rates between 10 and 100 pages per second as representative) due to the complex filtering and classification steps employed in our setup. This sums up to more than 80 days of pure crawling and classification for downloading and analyzing approx. 21 million web pages. The crawl yielded 373 GB presumably relevant and 607 GB presumably irrelevant pages, which corresponds to a harvest rate of 38%. This seems to be a typical value for such systems (e.g., [3, 22] report harvest rates between 25% and 45%). Document pre-selection was very effective: MIME-type filtering decreased the number of documents to be analyzed by 9.5%, language filtering by 14%, and document length filtering by 17%.

Evaluating a focused crawler is notoriously difficult for multiple reasons. First, experiments cannot be repeated due to the highly dynamic nature of the web. Starting a crawl with exactly the same set of seeds will result in a largely different result even if the repetition is performed shortly after the first run, as many pages will have changed leading to different link chains. Due to this fact, one cannot easily compare the performance of, for instance, two crawls using different classifiers, different relevance thresholds, or just different prioritization rules for the fetch queue. Second, the recall of a crawler cannot be determined; even estimating it is impossible as this would require a certain set of pages one expects to be found; but whether or not a crawler finds them largely depends on the seeds which cannot be set in an "unbiased" fashion. Third, yield and harvest rate depend largely on the seed lists, which usually are not published.

Classifier and Boilerplate Detection. The quality of the crawled corpus for our purposes, i.e., its specificity for the biomedical domain, depends on mostly two factors: The quality of the classifier, and the quality of the boilerplate detection. We assessed the quality of both components on a gold standard data set during development and on a small, randomly drawn sample of the crawl.

Our classifier achieved a precision of 98% at a recall of 83% in 10-fold cross validation on its training corpus. We then manually checked a randomly drawn set of 100 pages from the relevant corpus and 100 pages from the irrelevant corpus. On these 200 pages, precision was estimated at 94% at a recall of 90%, which roughly confirms the results on the training data (note that these are quality measures of the classifier, not of the the entire crawler; see discussion

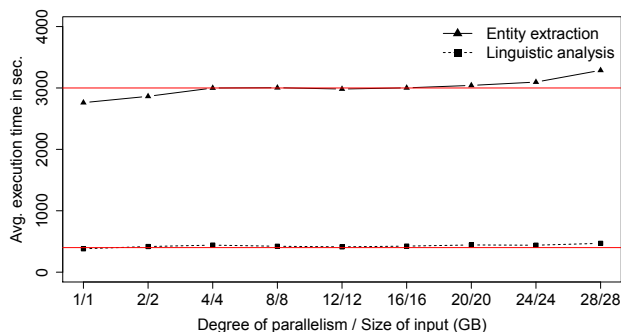


Figure 4: Scale up of linguistic and entity extraction partial data flows. Ideal scale up is displayed in red.

above). Differences are notable, but in expectable ranges gives the different characteristics of the texts and the small sample size. An analysis of the false positives showed that these are often web pages at the fringe of what we consider biomedical; for instance, pages describing chemical support for body builders or technical devices used for medical purposes such as wheel chairs. Note that the classifier model we used is geared towards high precision as classifier recall plays a minor role in focused crawling; assuming that the web is essentially infinite, one can simply let the crawler run for longer to obtain more relevant documents. Whether this assumption was reasonable will be discussed in Sect. 5.

In an initial evaluation on a gold standard data set, the boilerplate detection tool we used achieved a precision of 90% at a recall of 82% on average, evaluated on a set of 1906 web pages. These quality measures are computed based on the amount of net text being correctly identified by the algorithm. We assessed the quality of the method on the same 200 web pages used for judging the text classifier. Results indicate a precision of 98% at a recall of 72%. Manual inspection revealed that tables and lists, which often contain valuable facts, are not recognized properly in many cases.

Table 2 lists the top 30 ranked domains according to page rank. Manual inspection revealed that many of them clearly relate to biomedical content, which suggests that the crawling process points to our target domain. Sub-classes of seemingly irrelevant sites, such as slideshare.net or blogger.com often also contain some biomedical material (e.g., blogs, personal journals, reports). It is also not surprising that domains such as arxiv.org and nature.com are ranked within the top 30, because seeds were generated by the search APIs of these domains, which return results only for content hosted there.

4.2 Scalability of Information Extraction

We evaluated the performance of the extraction and analysis data flow in terms of scalability and performance of the individual IE components. These experiments were carried out on a 28 node cluster, where each node was equipped with 24 GB RAM, 1 TB HDD, and a Intel Xeon E5-2620 CPU with 6 cores. Accordingly, the maximum degree of parallelism (DoP) was 168. In the following, we always report as runtimes the average of 3 runs of the analysis flows on each corpus. Input and output of all tasks was stored in HDFS

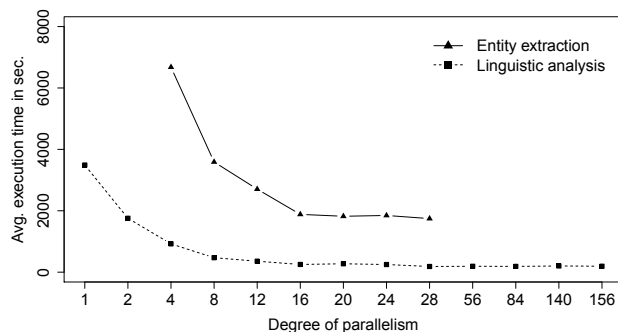


Figure 5: Scale out of linguistic and entity extraction data flows.

with one data node per compute node and a data replication factor of 3.

Runtime characteristics of the different IE tools.

All NLP and IE tools available for Stratosphere were originally designed and implemented by third parties. Many of them are complex applications encompassing several thousand lines of code with multiple dependencies to external libraries. This implies that we usually have no influence on the speed or memory consumption of these tools; there are very rare cases where command-line parameters can be used that impact these properties. Of course, we do heavily influence the speed of each tool on the entire data set by parallelizing its execution on different partitions of the data set.

Prior to analyzing the entire data set of crawled documents, we first evaluated the individual runtimes of each involved component using a random sample of 10,000 documents, which were analyzed using a single thread on a single server. The two dominant steps with respect to runtime are entity extraction, consuming 70% of the total execution time, and part-of-speech tagging, requiring 12% of the runtime. The distribution of the runtimes of part-of-speech tagging (a) and entity annotation (b) by sentences are shown in Fig. 3. Two observations are particularly interesting. First, our part-of-speech tagger, MedPost, uses a Hidden Markov Model of order three, whose runtime is, in principle, linear in the length of the text being analyzed. There are, however, large runtime fluctuations in practice (see Fig. 3(a)) and even occasional crashes, especially when the tagger is applied to very long sentences. Clearly, it is highly questionable whether the very long sentences we observe in our data (with more than 2000 characters) are really reasonable sentences or just errors of the sentence detection method; however, such errors are inevitable in a web environment, considering that the input to the splitter are parts, possibly wrongly extracted by the boilerplate detection, of arbitrary web pages possibly without any sentence structures (see also Sect. 5). One work-around would be to introduce an upper limit on sentence length, but finding a good threshold, trading runtime robustness for information yield, will be non-trivial. Second, Fig. 3(b) shows that the execution time needed for annotating entities varies greatly between annotation methods. Dictionary- and ML-based methods differ in runtime by up to three orders of magnitude. This is a consequence of the differing computational complexity

of the underlying algorithms; essentially linear for dictionary matches (the regular expression transformations almost only affect very short word suffixes), yet quadratic for the Conditional Random Fields underlying our ML-based tools [28].

Scalability. We tested the scalability of our IE data flow using a random sample of 20 GB from our crawl. Experiments were carried out separately for the linguistic analysis and the biomedical entity annotation to gain insights into their specific behavior. To this end, we created two separate data flows. Both first filter long texts, repair and remove HTML markup, and annotate sentence and token boundaries (cf. Fig. 2). Subsequently, the linguistic data flow detects pronouns, negation, and parenthesis, while the entity extraction flow first annotates part-of-speech tags and then drug, gene, and disease names using either dictionary or ML-based tools.

We first evaluated both flows on the 20 GB sample with varying DoPs, which led to a number of interesting observations. First, we could not execute the entity extraction data flow with a DoP smaller than 4 due to the excessive runtimes of the ML-based taggers (see above). Furthermore, we could not run this flow with DoPs larger than 28 due to the very high memory requirements of the dictionary-based taggers which each require between 6 and 20 GB of main memory per worker thread. Very likely, this is due to the fact that they transform each dictionary entry (i.e., a regular expression) into a the corresponding non-deterministic finite automaton, which usually greatly increases space requirements. However, the nodes we used have only 24 GB main memory; thus, we could not run more than one instance of these tools per node in the cluster. In contrast, the much less demanding linguistic data flow could be scaled out over the entire range of DoPs without any problems.

As shown in Fig. 5, scale out for both tested flows was satisfactory until DoP=16 for entity extraction, with a decrease in execution time of up to 72%, and until DoP=12 for the linguistic analysis, with a decrease in execution time of up to 95%. Using more nodes brought only marginal further improvements in execution times. This behavior can be explained by the relatively high start-up times of certain tools. For instance, the dictionary-based gene name recognition algorithm needs approximately 20 minutes (!) to load the dictionary and to create the internal data structures used for text matching. These 20 minutes are a hard lower bound for the runtime of this task, regardless of the number of nodes being used. It is not possible to work around this bound in a non-intrusive manner; one either has to use another tool or perform substantial changes to the tool itself. Scale-out of the linguistic flow was considerably better because in this data flow, startup costs of all involved tasks are negligible.

Clearly, the concrete DoP beyond which no more performance gains are obtained depends on the size of the input data, which was rather small in our scale-out experiments. Therefore, we also performed scale-up experiments, where we increased the number of available compute nodes synchronously to the amount of input data. As can be seen from Fig. 4, the linguistic data flow exhibits an almost ideal scale-up, whereas the entity extraction flow scales sub-linear for large DoPs and input sizes, which is consistent to the result of the scale-out experiments.

Processing the entire crawl - a war story. Although Stratosphere offers an elegant and powerful way of specifying complex IE data flows and is also capable of optimizing

about.com	arxiv.org
bettermedicine.com	biomedcentral.com
blogs.nature.com	blogger.com
cancer.net	cancer.org
cdc.gov	definition-of.com
disqus.com	farlex.com
g2conline.org	healthline.com
hhs.gov	lexiophiles.com
mpg.org	mypacs.net
nih.gov	omniture.com
ourhealth.com	reuters.com
rightdiagnosis.com	sideeffects.embl.de
slideshare.net	statcounter.com
thefreedictionary.com	wikimedia.org
wikipedia.org	wordpress.org

Table 2: Domains of 30 top-ranked sites according to page rank.

and parallelizing them on the given cluster, we could not execute the complete flow on the available hardware. This had mostly three reasons:

First, as described above, the dictionary-based entity taggers come along with very high main memory requirements. The complete data flow as shown in Fig. 2 needs roughly 60 GB main memory per worker thread, which clearly exceeds the amount of RAM available on each node. Unfortunately, the scheduling component of Stratosphere does not consider memory consumption per worker node as optimization goal, for that reason we could not run an entire flow on any of the nodes.

Second, another issue occurred with the ML-based disease recognition tool. This tool brings its own linguistic preprocessing, which is imported from the OpenNLP library, version 1.4¹². However, all other OpenNLP operators we integrated into Stratosphere (such as tokenization and sentence splitting) are based on version 1.5, which is not downward compatible to 1.4. Unfortunately, the Java class loader employed in the system’s runtime engine is not capable of using two different versions of the same library.

Third, in most popular Big Data analysis scenarios, a very large input is scanned and subsequently aggregated into smaller and smaller intermediate results. Accordingly, the bulk of the network traffic for accessing and inputs and writing outputs of tasks is usually spent by those tasks that are at the beginning of the analysis flows, whereas latter tasks often process only small data sets. However, the situation is quite different in a text analytics scenario like ours, where texts are piped through a series of tasks, each adding specific annotations (POS tags, entity annotation, token boundaries etc.) and thus actually increasing the size of the data throughout the analysis pipeline. In our case, the total amount of data produced is 1,6 TB, consisting of 400 GB entity annotations and 1,2 TB linguistic annotations, on top of the 1 TB raw text. Storing and accessing these large intermediate data sets through HDFS over-stressed the cluster network (nodes are connected by a 1 GB switch), leading to unpredictable network delays which in turn led to time-out induced crashes in some of the annotation tools.

To cope with these problems, we had to take several dras-

¹²<http://opennlp.apache.org/>

	Size in GB	No. of docs.	Mean no. of chars.
<i>Relevant crawl</i>	373	4,233,523	88,384
<i>Irrelevant crawl</i>	607	17,704,365	37,625
<i>Medline</i>	21	21,686,397	865
<i>PMC</i>	19	250,440	55,704

Table 3: Summary of data sets.

tic measures. First, we split up the flow into different parts such that each part only required memory within the given limits; essentially, we created one flow for all linguistic analysis and one flow per entity class of the biomedical analysis. Still, the memory requirements (see above) put severe constraints on the number of threads executable per node, which grossly hampered the overall DoP and thus greatly increased the overall runtime. Eventually, we spinned off gene recognition, being the most space-consuming task, and executed it on a single server with 1 TB RAM using 40 threads. In the same manner, we had to perform disease name extraction in a separate run to overcome versioning problems. To cope with the network problems, we furthermore splitted the crawled data into chunks of 50 GB and executed the different flows separately on these chunks.

Alternatively to splitting the data flow into separate parts and analyzing the crawled data chunk-wise, we considered renting cloud systems for our experiments but quickly disregarded this option due to high rental costs. Since memory requirements of IE operators is the most limiting factor in our study, only cloud instances with more than 35 GB RAM come into question, which are still rather costly. For example, renting a 28 node cluster of instance type "m4.4xlarge" with 64GB of RAM available per node from AWS costs approximately 650 USD per day as of March 2016¹³.

4.3 Content analysis

A main interest in this project is to compare extraction results from the web to the scientific literature to identify commonalities and differences between biomedical texts from the web and from peer reviewed journals. Thus, we also analyzed abstracts and full-texts from Medline and PMC using the same IE flow (downstream from the HTML treatment) as for the crawled data. As a forth text collection, we used all pages crawled but classified as irrelevant. Table 3 summarizes the data sets enclosed in this analysis.

We performed an in-depth analysis and comparison of the results obtained on these four corpora. Our analysis is split into a linguistic part, concerning properties such as article lengths, sentence structure, and usage of grammatical structures, and a domain-specific part, comparing the occurrence frequencies and distributions of three biomedical entities, i.e., drugs, genes, and diseases.

4.3.1 Linguistic analysis

Analyzing the linguistic structure of texts is important for assessing the complexity of texts and to judge whether existing IE tools, which were trained and developed for different corpora, might perform well in web documents. Particularly, we examine document lengths, sentence lengths, incidence of negation, and incidence of pronouns and parenthesis. Dif-

ferences in obtained measures were statistically assessed using the Mann-Whitney-Wilcoxon signed rank test. This test produces a P-value, which estimates the probability that the observed differences are due to random effects in the data.

Document and sentence lengths. Sentence lengths impact IE and NLP in different ways. First, the execution time of IE and NLP tools usually directly depends on the lengths of the sentences to be analyzed. Second, the difficulty of constituent and dependency sentence parsing and the difficulty of modern relation extraction methods rises with sentence lengths [27]. Likewise, if crawled web documents contain shorter sentences than Medline or PMC, we expect the former to be easier to analyze. Other important measures are the document lengths in the different corpora, as these must be considered when comparing the frequency of entity mentions.

Figure 6(a) displays the distribution of document lengths and Fig. 6(b) displays mean sentence lengths across the different data sets. Mean document lengths in relevant documents were significantly shorter than in PMC ($P < 0.01$), but significantly longer compared to irrelevant documents ($P < 0.01$) and to Medline abstracts ($P < 0.01$). Document lengths for the relevant corpus show the largest variance, which increases the need for appropriate load balancing in a distributed setting. Differences in sentence lengths between the four corpora are also significant, confirming previous findings from [6] regarding Medline abstracts and PMC full texts. These differences have to be kept in mind when selecting tools for IE that are based on gold standard data. Most tools we are aware of were trained and evaluated on Medline abstracts and thus on rather short sentences; accordingly, we expected a lower performance of these tools on longer sentences than reported in the literature.

Incidence of negation. Detecting negation is important in many areas of natural language processing (e.g., sentiment analysis, relation extraction) and is particularly important for analyzing biomedical texts [1]. Here, we used a rather simple method for determining negations in sentences, using a set of regular expressions to find mentions of the words *not*, *nor*, and *neither*. As shown in Fig. 6(c), the incidence of negation in the four corpora is significantly different ($P < 0.01$) regarding the overall incidence of negation and the relative frequency of negation with respect to document length. Specifically, texts in the set of relevant documents have a lower incidence of negation than in PMC and the irrelevant pages and a higher incidence of negation than in Medline. Accordingly, appropriate treatment of negation will be more important for web data than for scientific articles.

Incidence of pronouns and parenthesized text. Pronominal anaphora are important in biomedical IE to perform co-reference resolution [10]. To measure the amount of such co-references in our corpora, we counted six different classes of pronouns in each data set. Interestingly, the incidence of demonstrative, relative, and object pronouns, which are the most important pronoun classes for co-reference resolution, was significantly lower both in relevant and irrelevant texts compared to texts from PMC (data not shown). We expected this observation concerning irrelevant texts since these texts are significantly shorter than texts from PMC. Our observation is surprising for relevant texts as these are significantly longer than documents from PMC. This finding

¹³<http://aws.amazon.com/ec2/pricing/>

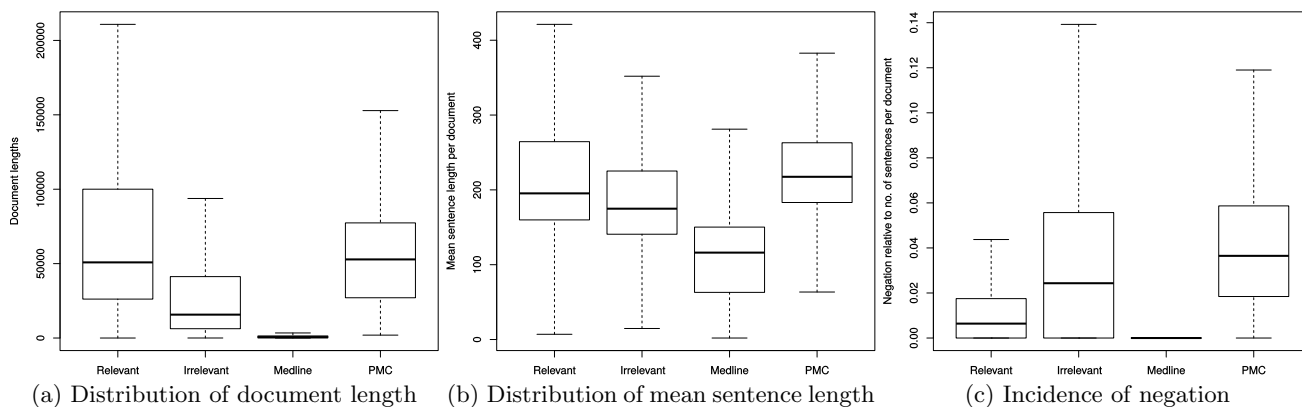


Figure 6: Distribution and incidence of linguistic properties per document in different data sets.

might indicate that co-reference resolution on crawled texts is not as vital as in analyzing biomedical full-text literature.

Parentheses can hint to abbreviations, paper references, synonyms of named entities, etc., which are very important during NLP processing. Properly treating parenthesis is also highly important for syntactic parsing, as text in parentheses does typically not conform to the sentence grammar. We extracted parenthesized text using a set of regular expressions and found that their incidences differ significantly ($P < 0.01$) between all data sets (data not shown). We observed the highest incidence in texts from PMC, followed by relevant web documents and Medline, and the lowest in irrelevant documents.

4.3.2 Biomedical entities

The result we were most interested in this study from an application point-of-view was the degree of differences in the biomedical entities that are mentioned on the web versus the scientific literature. Table 4 lists the number of distinct names that were found in the two crawled corpora, in Medline, and in PMC for the entity classes disease, drug, and gene. As expected, ML-based annotation produces substantially more annotations for each entity type than dictionary-based annotation. We also notice that the total number of distinct annotations is significantly different between relevant and irrelevant pages both for dictionary- and ML-based approaches, with much larger numbers of annotations in relevant documents for each entity class, which is reassuring of the crawl quality. In the following, we compare named entity annotations in the different corpora more deeply for each entity type.

Disease names. As shown in Fig. 7(a), the incidence of disease names per document is higher in PMC articles and relevant web documents compared to Medline abstracts and irrelevant crawled documents, which rarely mention more than one disease (presumably mostly false positives with abbreviations). Differences of the mean number of disease annotations per 1000 sentences between relevant ($avg_{rel} = 128.49$) and irrelevant ($avg_{irrel} = 4.57$), relevant and Medline ($avg_{medl} = 204.92$), and Medline and PMC ($avg_{pmc} = 117.51$) are all highly significant ($P < 0.01$). One explanation for the smaller incidence of disease names in Medline abstracts compared to web and PMC full-text documents even when normalized to per-sentence measures is the shorter average sentence length in the abstracts. Fur-

Data set	Method	Disease	Drug	Gene
Relevant	Dict.	26,344	17,974	73,435
	ML	629,384	28,660	5,506,579
Irrelevant	Dict.	5,318	8,456	22,131
	ML	119,638	15,875	991,010
Medline	Dict.	11,194	12,164	29,928
	ML	343,184	20,282	4,715,194
PMC	Dict.	12,291	15,013	92,319
	ML	277,211	25,462	1,858,709

Table 4: Number of distinct entity names by corpus.

thermore, the large number of disease names in the relevant web document is certainly a result of the way we generated seeds, often using disease names as keywords. Accordingly, our crawl should be enriched for disease-related websites, such as information websites for patients or disease-specific support groups.

Drug names. Figure 7(b) displays the incidence of drug names per document in the different data sets. In relevant documents, more drug mentions were recognized on average compared to irrelevant documents and abstracts taken from Medline. The means of drug name annotations for both annotation methods combined per 1000 sentences differ significantly ($P < 0.01$) between relevant ($avg_{rel} = 97.83$) and irrelevant ($avg_{irrel} = 6.85$) documents, and between relevant documents and Medline ($avg_{medl} = 293.95$) abstracts and PMC documents ($avg_{pmc} = 275.95$). Possible reasons for the differences between crawled relevant documents and Medline abstracts are the same as for disease names, as disease-related web sites often are also full of drug names.

Gene names. Similar to results for disease and drug names, differences in the number of gene mentions per document are significant between relevant and irrelevant documents, and between relevant and Medline ($P < 0.01$) and the means of gene name annotations per 1000 sentences for dictionary-based annotation differ significantly between all sets ($avg_{rel} = 128.23$, $avg_{irrel} = 4.39$, $avg_{medl} = 415.58$, $avg_{pmc} = 74.12$). However, the most striking observation regarding the incidences of gene names are the extremely large differences between the dictionary-based extraction method

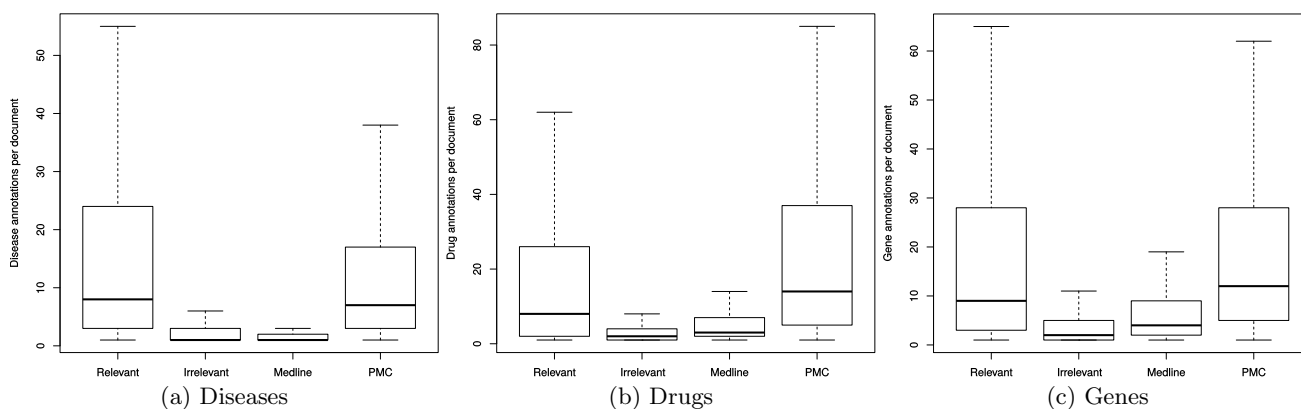


Figure 7: Incidence of named entity annotations per document in the different corpora.

and the ML-based algorithm. Using ML, we recognized more than 5.5 million distinct gene names in relevant documents, whereas dictionary-based annotation only finds 73,435 different gene names (cf. Table 4). It is immediately clear that the vast majority of annotations produced by BANNER must be false positives, because there exist only roughly 900,000 distinct gene names (including synonyms) in the public gene-related databases. The reason for this presumably excessive amount of wrong annotations is the fact that BANNER was trained on a small set of selected Medline abstracts, which exhibit different language characteristics than web documents (see Sect. 4.3.1). Upon manual inspection, we noticed that a very large number of false positives are three letter acronyms (TLA), which are almost always tagged as genes by our tool; this strategy is correct for the gold standard abstracts used for developing and evaluating the tool, but leads to catastrophic performance on any other documents. Therefore, we filtered all TLAs from the list of ML-tagged gene names prior to further analysis, reducing, for instance, the number of distinct gene names in the relevant web corpus from 5.5 million to 2.3 million. Figure 7(c) displays the incidence of gene names in the different data sets after filtering.

Annotation overlap and difference. Finally, we determined the differences in the sets of extracted entities between the different data sets to assess whether focused crawling has the potential to open up new sources of biomedical knowledge and to exclude the potential danger of simply having many scientific abstracts in the relevant crawl. Figure 8 shows the overlap and non-overlaps of dictionary-based entity annotations. For all evaluated entity types, we found that the overlap of extracted names between relevant and irrelevant documents is notable but rather small, i.e., approximately 15% for disease names, approximately 30% for drug names (86% out of which were also found in Medline and PMC), and 17% for gene names. The overlap between relevant and Medline and relevant and PMC is considerably larger and ranges from 6% (ML-based gene extraction) to 60% (dictionary-based gene extraction).

We also assessed the statistical significance of these differences using the Jensen-Shannon divergence (JSD), an information-theoretic measure for assessing the difference between two probability distributions based on the Kullback-Leibler divergence. JSD is a symmetric measure and results in values bounded for two distributions A, B with $0 \leq$

$JSD_{A,B} \leq 1$, where similar distributions approach a JSD close to 0, and dissimilar distributions approach 1. For each data set and entity type, we determined the probability distribution of entity names and computed for each combination of data sets the JSD (data not shown). Relevant and irrelevant documents exhibit larger JSDs, i.e., significantly different distributions ($0.4463 \leq JSD_{rel,irrel} \leq 0.6548$), for any entity type compared to documents from Medline and relevant ($0.2864 \leq JSD_{rel,medl} \leq 0.3596$), and PMC and relevant ($0.1673 \leq JSD_{rel,pmc} \leq 0.3354$). This indicates that documents classified as relevant during crawling actually are more similar to the biomedical literature and thus relevant for the target domain. Similarly, JSD between irrelevant and Medline ($0.4528 \leq JSD_{irrel,medl} \leq 0.6850$) and irrelevant and PMC ($0.3941 \leq JSD_{irrel,pmc} \leq 0.6633$).

Together, these findings give clear evidence that there is a significant amount of information on the web which is not contained in the scientific literature, indicated by several thousand distinct entity names for each entity type which appear only in relevant web documents. Studying these sets in more detail will be the next step in our research.

5. LESSONS LEARNED AND CHALLENGES FOR FUTURE RESEARCH

In this paper, we reported our experiences with a study in crawling and deeply analyzing a large domain-specific corpus from the web, exemplified for the field of biomedical research. From a domain-knowledge point-of-view, our results are highly interesting as they indicate that there is indeed a large body of biomedical knowledge on the web that is not present in the scientific literature. Clearly, much more research is necessary to substantiate this hypothesis and to assess the usefulness of this knowledge, which could be, for instance, reports of high quality that were not (yet) published or important text book knowledge that is so established that one cannot find scientific publications discussing it; however, a large fraction presumably are also false positive matches of the taggers or information of dubious quality and reliability. Our study also brought up a number of technical pitfalls of focused crawling and large-scale, high quality text processing using a best-of-breed strategy for choosing text analytics tools. In the following, we highlight some of these challenges.

Reliable MIME-type detection. Large files down-

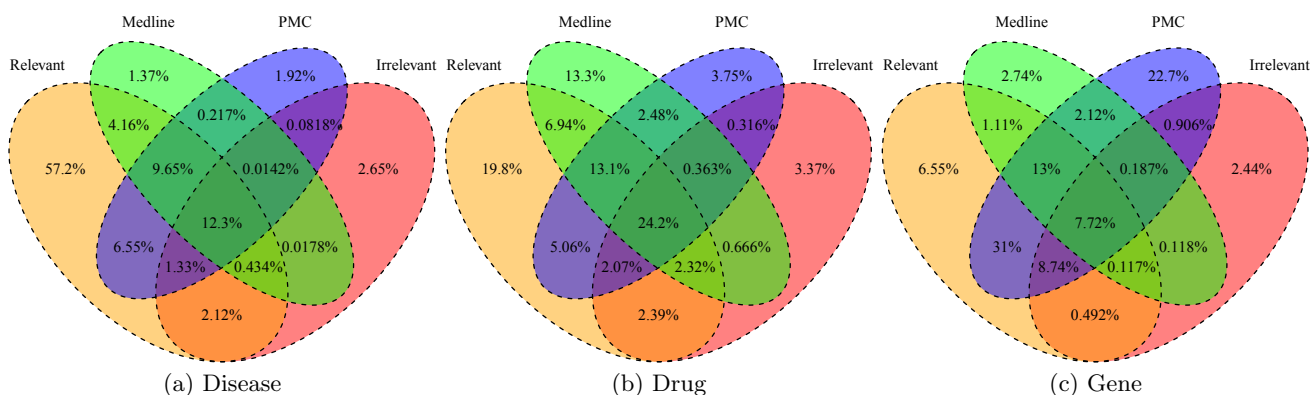


Figure 8: Annotation overlap of distinct entity names in % for different entity types and dictionary-based annotation.

loaded during crawl are often not textual but embedded presentation slides or formatted documents, which were wrongly classified as plain textual. Filtering by document size only, as we did, is unsatisfactory as it easily misses relevant (and extensive) content. However, we are not aware of any robust tools or ongoing research for reliable MIME-type detection; instead, detecting MIME-types usually is carried out by regular expression matching on the file name extension or by analyzing the first n bytes of a document. We used the Apache Tika¹⁴ library during crawling, which ships only with a list of a handful common MIME-types.

Robust HTML markup removal. According to [19], 95% of HTML documents on the web do not adhere to W3C HTML standards. 13% of the analyzed websites had so severe issues that they could not be transcoded. However, correctly formatted HTML pages seem to be a prerequisite for most boilerplate detection algorithms. In the course of this study, we evaluated different markup removal algorithms and found them to perform reasonably well on a gold-standard set of 1,906 pages (details omitted). Applying these tools to our crawled documents, however, revealed that they are highly sensitive to markup errors, often resulting in crashes or empty results. Developing boilerplate detection algorithms that are more robust against errors in real life web pages is essential for seamless and comprehensive text analytics from web documents.

Resource consumption of extraction tools. Even though the NLP and IE tools we used require only a moderate amount of memory for each running instance, these numbers sum-up notably when running multiple instances on a multi-threaded machine. In our study, this grossly hampered the DoP we could achieve, leading to sub-optimal resource usage and long analysis times. Furthermore, several tools produced Java out-of-memory errors when applied to long texts; as we could not find a workaround for this behavior, we eventually had to define a hard upper limit on the texts to be analyzed. These findings indicate that the current tools, at least in the biomedical domain, are not well prepared to be applied on large text collections. This is consistent with the observation that research in this field is obsessed with small improvements in extraction quality on small gold standard data sets, and rarely considers runtime or space consumption when used in true applications.

The situation might be different in other fields, but research in more robust NER tools, with configurable memory consumption, seems highly desirable.

NLP and IE models for web documents. It is well known that ML tools work best on data sets that exhibit similar language characteristics as those used for training. Most research into NER tools in biomedicine is performed on Medline abstracts, both with respect to training data and evaluation data. On such data, ML-based NER is clearly superior to other approaches, as shown in many recent studies and international competitions [25]. Accordingly, all ML-based methods used in this project employ models trained on Medline abstracts since no other training data is available. However, our study reveals that web documents and documents from Medline and PMC are significantly different in several aspects. This leads to an enormous amount of false positive matches by these tools, which are often short abbreviations. We believe that there is a great need for more sophisticated models for domain-specific entity recognition from web documents. Note that the current research into this direction typically targets rather simple entity types, such as persons, places, or products [9]. To our knowledge, the performance of such methods on the more difficult biomedical entity types has not yet been evaluated.

Trade-off between precision and yield in focused crawling. When setting-up our system, we focused on a high-precision text classifier as we believed that the number of true positives can be improved more easily with longer crawls than with a high-recall classifier, which might also bring many false positive pages. However, we actually observed that this strategy was not as effective as we thought. Actually, the size of the crawl we obtained was bound by the fact that our crawl frontier eventually emptied. As described in Sect. 2.2, we already had to significantly extend our seed list to obtain a crawl of at least the size we have now. Several strategies could be followed to create larger focused crawl. For instance, one could produce even larger seed lists, but this requires substantial preparation time given the current limits of the search engine APIs. Another approach would be to also follow links from pages classified as irrelevant, but only with a small margin. Finally, one could tune the classifier towards more recall during crawling, and classify each crawled text later a second time with a model geared

¹⁴<http://tika.apache.org>

towards high precision. Which of these ways is the most promising one, remains an open question.

Crawling and text analytics as a consolidated process. This project pursued a two-staged approach, where crawling and text analytics was performed in two separate phases using very different infrastructures. However, the result of the IE pipeline could actually be a valuable input for the classifier during a crawl, as the occurrence of gene names or disease names are strong indicators for biomedical content. We believe it would be a worthwhile undertaking to research systems that would allow specifying crawling strategies, classification, and domain-specific IE in a single framework. Such a framework would greatly reduce the time it takes to build web-scale domain-specific text analysis systems.

6. CONCLUSION

In recent years, web-scale information extraction by means of Big Data processing tools has gained much attention. However, most works focus on comparable simple IE techniques [26, 16] or consider only singular extraction tasks [4, 14]. Moreover, we are not aware of any research has in the interplay of web crawling and state-of-the-art information extraction, especially when applied in a domain-specific setting. Accurate acquisition of web documents by means of focused web crawls was in the focus of research in the late 1990s and early 2000s, but this topic seems to have lost momentum in recent years, despite the often reiterated importance of the web for Big Data scenarios. We can only speculate about the reasons for this apparent contradiction. One reason could be the difficulty to compare and assess the quality of focused web crawling; another reason probably is the fact that such research requires considerable investments both in terms of hardware (large clusters, high bandwidth, etc.) and in terms of software (complex analysis pipelines, best-of-breed tool selection, parallelization).

We described methods to simplify the creation of such systems using and extending open source systems. The experiences we report partly seem devastating, uncovering a series of issues in essentially all components that prevent their seamless and scalable application, at least on a mid-size cluster as was available to us. Of course, our experiences are specific to the frameworks we used and might have been different with other base systems; yet, we are not aware of any other system offering such a rich set of NLP and (biomedical) IE tools ready-to-use (with all caveats described in this work) than Stratosphere. Nevertheless, it is our frank intention to create more attention to the interesting and challenging topic of sophisticated, domain-specific web-scale information extraction in the data management research community.

Acknowledgments

This research is funded by the German Research Foundation under grant FOR 1306. We thank Philippe Thomas and Tim Rocktäschel for providing models and software we used during entity recognition.

7. REFERENCES

- [1] S. Agarwal and H. Yu. Biomedical negation scope detection with conditional random fields. *J. Am. Med. Inform. Assn.*, 17(6):696–701, 2010.
- [2] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *VLDB J.*, 23(6):939–964, 2014.
- [3] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Comput. Netw.*, 31(11-16):1623–1640, 1999.
- [4] B. Chandramouli, J. Goldstein, and S. Duan. Temporal analytics on big data for web advertising. In *Proc. 28th IEEE Int. Conf. on Data Engineering*, pages 90–101, 2012.
- [5] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: An algebraic approach to declarative information extraction. In *Proc. 48th Annual Meeting of the Association for Computational Linguistics*, 2010.
- [6] K. B. Cohen, H. L. Johnson, K. Verspoor, C. Roeder, and L. Hunter. The structural and content aspects of abstracts versus bodies of full text journal articles are different. *BMC Bioinformatics*, 11:492, 2010.
- [7] L. Covolo, S. Mascaretti, A. Caruana, G. Orizio, L. Caimi, and U. Gelatti. How has the flu virus infected the web? 2010 influenza and vaccine information available on the internet. *BMC Public Health*, 13(1):83, 2013.
- [8] B. D. Davison. Topical locality in the web. In *Proc. 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 272–279, 2000.
- [9] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, 2005.
- [10] C. Gasperin and T. Briscoe. Statistical anaphora resolution in biomedical texts. In *Proc. 22nd Int. Conf. on Computational Linguistics - Volume 1*, pages 257–264, 2008.
- [11] M. Gerner, G. Nenadic, and C. M. Bergman. Linnaeus: A species name identification system for biomedical literature. *BMC Bioinformatics*, 11:85, 2010.
- [12] A. Heise and F. Naumann. Integrating open government data with stratosphere for more transparency. *J. Web Semant.*, 14:45–56, 2012.
- [13] A. Heise, A. Rheinländer, M. Leich, U. Leser, and F. Naumann. Meteor/Sopremo: An Extensible Query Language and Operator Model. In *Proc. Int. Ws. on End-to-End Management of Big Data*, 2012.
- [14] V. N. Khuc, C. Shivade, R. Ramnath, and J. Ramanathan. Towards building large-scale distributed systems for twitter sentiment analysis. In *Proc. 27th Annual ACM Symposium on Applied Computing*, pages 459–464, 2012.
- [15] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proc. 3rd ACM Int. Conf. on Web Search and Data Mining*, pages 441–450, 2010.
- [16] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. Systemt: A system for

- declarative information extraction. *SIGMOD Rec.*, 37(4), 2009.
- [17] R. Leaman and G. Gonzalez. Banner: An executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*, pages 652–663, 2008.
- [18] U. Leser and J. Hakenberg. What makes a gene name? named entity recognition in the biomedical literature. *Brief. Bioinform.*, 6(4):357–369, 2005.
- [19] E. Ofuonye, P. Beatty, S. Dick, and J. Miller. Prevalence and classification of web page defects. *Online Inform. Rev.*, 34(1):160–174, 2010.
- [20] C. Olston and M. Najork. Web crawling. *Found. Trends Inf. Retr.*, 4(3):175–246, 2010.
- [21] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2008.
- [22] G. Pant and P. Srinivasan. Learning to crawl: Comparing classification schemes. *ACM T. Inform. Syst.*, 23(4):430–462, 2005.
- [23] A. Rheinländer, A. Heise, F. Hueske, U. Leser, and F. Naumann. Sofa: An extensible logical optimizer for udf-heavy data flows. *Inform. Syst.*, 52(C):96–125, 2015.
- [24] T. Rocktäschel, M. Weidlich, and U. Leser. Chemspot: a hybrid system for chemical named entity recognition. *Bioinformatics*, 28(12):1633–1640, 2012.
- [25] I. Segura-Bedmar, P. Martínez, and M. Herrero-Zazo. Lessons learnt from the ddiextraction-2013 shared task. *J. Biomed. Inform.*, 51:152–164, 2014.
- [26] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, pages 1033–1044, 2007.
- [27] D. Tikk, I. Solt, P. E. Thomas, and U. Leser. A detailed error analysis of 13 kernel methods for protein-protein interaction extraction. *BMC Bioinformatics*, 14:12, 2013.
- [28] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE T. Inform. Theory*, 13(2):260–269, 2006.
- [29] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 296–305, 2003.