Interactive Extractive Search over Biomedical Corpora

Hillel Taub-Tabib[1] **Micah Shlain**[1,2] **Shoval Sadde**[1] **Dan Lahav**[3]
Matan Eyal[1] **Yaara Cohen**[1] **Yoav Goldberg**[1,2]
[1] Allen Institute for AI, Tel Aviv, Israel
[2] Bar Ilan University, Ramat-Gan, Israel
[3] Tel Aviv University, Tel-Aviv, Israel
{hillelt,micahs}@allenai.org

Abstract

We present a system that allows life-science researchers to search a linguistically annotated corpus of scientific texts using patterns over dependency graphs, as well as using patterns over token sequences and a powerful variant of boolean keyword queries. In contrast to previous attempts to dependency-based search, we introduce a light-weight query language that does not require the user to know the details of the underlying linguistic representations, and instead to query the corpus by providing an example sentence coupled with simple markup. Search is performed at an interactive speed due to efficient linguistic graph-indexing and retrieval engine. This allows for rapid exploration, development and refinement of user queries. We demonstrate the system using example workflows over two corpora: the PubMed corpus including 14,446,243 PubMed abstracts and the CORD-19 dataset[1], a collection of over 45,000 research papers focused on COVID-19 research. The system is publicly available at https://allenai.github.io/spike

1 Introduction

Recent years have seen a surge in the amount of accessible Life Sciences data. Search engines like Google Scholar, Microsoft Academic Search or Semantic Scholar allow researchers to search for published papers based on keywords or concepts, but search results often include thousands of papers and extracting the relevant information from the papers is a problem not addressed by the search engines. This paradigm works well when the information need can be answered by reviewing a number of papers from the top of the search results. However, when the information need requires extraction of information nuggets from many papers (e.g. *all chemical-protein interactions* or *all risk factors for a disease*) the task becomes challenging and researchers will typically resort to curated knowledge bases or designated survey papers in case ones are available.

We present a search system that works in a paradigm which we call Extractive Search, and which allows rapid information seeking queries that are aimed at extracting facts, rather than documents. Our system combines three query modes: boolean, sequential and syntactic, targeting different stages of the analysis process, and different extraction scenarios. Boolean queries (§4.1) are the most standard, and look for the existence of search terms, or groups of search terms, in a sentence, regardless of their order. These are very powerful for finding relevant sentences, and for co-occurrence searches. Sequential queries (§4.2) focus on the order and distance between terms. They are intuitive to specify and are very effective where the text includes "anchor-words" near the entity of interest. Lastly, syntactic queries (§4.4) focus on the linguistic constructions that connect the query words to each other. Syntactic queries are very powerful, and can work also where the concept to be extracted does not have clear linear anchors. However, they are also traditionally hard to specify and require strong linguistic background to use. Our systems lowers their barrier of entry with a specification-by-example interface.

Our proposed system is based on the following components.

Minimal but powerful query languages. There is an inherent trade-off between simplicity and control. On the one extreme, web search engines like Google Search offer great simplicity, but very little control, over the exact information need. On the other extreme, information extraction pattern-specification languages like UIMA Ruta offer great precision and control, but also expose a low-level

[1] https://pages.semanticscholar.org/coronavirus-research

Proceedings of the BioNLP 2020 workshop, pages 28–37

view of the text and come with over hundred-page manual.[2]

Our system is designed to offer *high degree of expressivity*, while remaining simple to grasp: the syntax and functionality can be described in a few paragraphs. The three query languages are designed to share the same syntax to the extent possible, to facilitate knowledge transfer between them and to ease the learning curve.

Linguistic Information, Captures, and Expansions. Each of the three query types are linguistically informed, and the user can condition not only on the word forms, but also on their lemmas, parts-of-speech tags, and identified entity types. The user can also request to *capture* some of the search terms, and to *expand* them to a linguistic context. For example, in a boolean search query looking for a sentence that contains the lemmas "treat" and "treatment" ('lemma=treat|treatment'), a chemical name ('entity=SIMPLE_CHEMICAL') and the word "infection" ('infection'), a user can mark the chemical name and the word "infection" as *captures*. This will yield a list of chemical/infection pairs, together with the sentence from which they originated, all of which contain the words relating to treatments. Capturing the word "infection" is not very useful on its own: all matches result in the exact same word. But, by *expanding* the captured word to its surrounding linguistic environment, the captures list will contain terms such as "PEDV infection", "acyclovir-resistant HSV infection" and "secondary bacterial infection". Running this query over PubMed allows us to create a large and relatively focused list in just a few seconds. The list can then be downloaded as a CSV file for further processing. The search becomes *extractive*: we are not only looking for documents, but also, by use of captures, *extract information* from them.

Sentence Focus, Contextual Restrictions. As our system is intended for extraction of information, it works at the sentence level. However, each sentence is situated in a context, and we allow secondary queries to condition on that context, for example by looking for sentences that appear in paragraphs that contain certain words, or which appear in papers with certain words in their titles, in papers with specific MeSH terms, in papers whose abstracts include specific terms, etc. This combines the focus and information density of a sentence, which is the main target of the extraction, with the rich signals available in its surrounding context.

Interactive Speed. Central to the approach is an indexed solution, based on (Valenzuela-Escárcega et al., 2020), that allows to perform all types of queries efficiently over very large corpora, while getting results almost immediately. This allows the users to interactively refine their queries and improve them based on the feedback from the results. This contrasts with machine learning based solutions that, even neglecting the development time, require substantially longer turnaround times between query and results from a large corpus.

2 Existing Information Discovery Approaches

The primary paradigm for navigating large scientific collections such as MEDLINE/PubMed[3] is document-level search.

The most immediate document-level searching technique is boolean search ("keyword search"). However, these methods suffer from an inability to capture the concepts aimed for by the user, as biomedical terms may have different names in different sub-fields and as the user may not always know exactly what they are looking for. To overcome this issue several databases offer semantic searching by exploiting MeSH terms that indicate related concepts. While in some cases MeSH terms can be assigned automatically, e.g (Mork et al., 2013), in others obtaining related concepts require a manual assignment which is laborious to obtain.

Beyond the methods incorporated in the literature databases themselves, there are numerous external tools for biomedical document searching. Thalia (Soto et al., 2018) is a system for semantic searching over PubMed. It can recognize different types of concepts occurring in Biomedical abstracts, and additionally enables search based on abstract metadata; LIVIVO (Müller et al., 2017) takes the task of vertically integrating information from divergent research areas in the life sciences; SWIFT-Review [4] offers iterative screening by reranking the results based on the user's inputs.

All of these solutions are focused on the document level, which can be limiting: they often surface hundreds of papers or more, requiring careful reading, assessing and filtering by the user, in order to locate the relevant facts they are looking for.

[2] https://uima.apache.org/d/ruta-current/tools.ruta.book.pdf

[3] https://www.ncbi.nlm.nih.gov/pubmed/

[4] https://www.sciome.com/swift-review/

To complement document searching, some systems facilitate automatic extraction of biomedical concepts, or patterns, from documents. Such systems are often equipped with analysis capabilities of the extracted information. For example, NaCTem has created systems that extract biomedical entities, relations and events.[5]; ExaCT and RobotReviewer (Kiritchenko et al., 2010; Marshall et al., 2015) take a RCT report and retrieve sentences that match certain study characteristics.

To improve the development of automatic document selection and information extraction the BioNLP community organized a series of shared tasks (Kim et al., 2009, 2011; Nédellec et al., 2013; Segura Bedmar et al., 2013; Deléger et al., 2016; Chaix et al., 2016; Jin-Dong et al., 2019). The tasks address a diverse set of biomed topics addressed by a range of NLP-based techniques. While effective, such systems require annotated training data and substantial expertise to produce. As such, they are restricted to several "head" information extraction needs, those that enjoy a wide community interest and support. The long tail of information needs of "casual" researchers remain mostly un-addressed.

3 Interactive IE Approach

Existing approaches to information extraction from bio-medical data suffer from significant practical limitations. Techniques based on supervised training require extensive data collection and annotation (Kim et al., 2009, 2011; Nédellec et al., 2013; Segura Bedmar et al., 2013; Deléger et al., 2016; Chaix et al., 2016), or a high degree of technical savviness in producing high quality data sets from distant supervision (Peng et al., 2017; Verga et al., 2017; Wang et al., 2019). On the other hand, rule based engines are generally too complex to be used directly by domain experts and require a linguist or an NLP specialist to operate. Furthermore, both rule based and supervised systems typically operate in a pipeline approach where an NER engine identifies the relevant entities and subsequent extraction models identify the relations between them. This approach is often problematic in real world biomedical IE scenarios, where relevant entities often cannot be extracted by stock NER models.

To address these limitations we present a system allowing domain experts to interactively query linguistically annotated datasets of scientific research papers, using a novel multifaceted query language which we designed, and which supports boolean search, sequential patterns search, and by-example syntactic search (Shlain et al., 2020), as well as specification of search terms whose matches should be captured or expanded. The queries can be further restricted by contextual information.

We demonstrate the system on two datasets: a comprehensive dataset of PubMed abstracts and a dataset of full text papers focused on COVID-19 research.

Comparison to existing systems. In contrast to document level search solutions, the results returned by our system are sentences which include highlighted spans that directly answer the user's information need. In contrast to supervised IE solutions, our solution does not require a lengthy process of data collection and labeling or a precise definition of the problem settings.

Compared to rule based systems our system differentiates itself in a number of ways: (i) our query engine automatically translates lightly tagged natural language sentences to syntactic queries (query-by-example) thus allowing domain experts to benefit from the advantages of syntactic patterns without a deep understanding of syntax; (ii) our queries run against indexed data, allowing our translated syntactic queries to run at interactive speed; and (iii) our system does not rely on relation schemas and does not make assumptions about the number of arguments involved or their types.

In many respects, our system is similar to the PropMiner system (Akbik et al., 2013) for exploratory relation extraction (Akbik et al., 2014). Both PropMiner and our system support by-example queries in interactive speed. However, the query languages we describe in section 4 are significantly more expressive than PropMiner's language, which supports only binary relations. Furthermore, compared to PropMiner, our annotation pipeline was optimized specifically for the biomedical domain and our system is freely available online.

Technical details. The datasets were annotated for biomedical entities and syntax using a custom SciSpacy pipeline (Neumann et al., 2019)[6], and the syntactic trees were enriched to BART format using pyBART (Tiktinsky et al., 2020). The annotated data is indexed using the Odinson engine (Valenzuela-Escárcega et al., 2020).

[5] http://www.nactem.ac.uk/

[6] All abstracts underwent sentence splitting, tokenization, tagging, parsing and NER using all the 4 NER models available in SciSpacy

4 Extractive Query Languages

4.1 Boolean Queries

Boolean queries are the standard in information retrieval (IR): the user provides a set of terms that should, and should not, appear in a document, and the system returns a set of documents that adhere to these constraints. This is a familiar and intuitive model, which can be very effective for initial data exploration as well as for extraction tasks that focus on co-occurrence. We depart from standard boolean queries and extend them by (a) allowing to condition on different linguistic aspects of each token; (b) allowing *capturing* of terms into named variables; and (c) allowing *linguistic expansion* of the captured terms.

The simplest boolean query is a list of terms, where each term is a word, i.e: 'infection asymptomatic fatal' The semantics is that all the terms must appear in the query. A term can be made optional by prefixing it with a '?' symbol ('infection asymptomatic ?fatal'). Each term can also specify a list of alternatives: 'fatal|deadly|lethal'.

Beyond words. In addition to matching words, terms can also specify linguistic properties: lemmas, parts-of-speech, and domain-specific entity-types: 'lemma=infect entity=DISEASE'. Conditions can also be combined: 'lemma=cause|reason&tag=NN'. We find that the ability to search for domain-specific types is very effective in boolean queries, as it allows to search for concepts rather than words. In addition to exact match, we also support matching on regular expressions ('lemma=/caus.*/'). The field names word, lemma, entity, tag can be shortened to w,l,e,t.

Captures. Central to our extractive approach is the ability to designate specific search term to be *captured*. Capturing is indicated by prefixing the term with ':' (for an automatically-named capture) or with 'name:' (for a named capture). The query 'fatal asymptomatic d:e=DISEASE' will look for sentences that contain the terms 'fatal' and 'asymptomatic' as well as a name of a disease, and will capture the disease name under a variable "d". Each query result will be a sentence with a single disease captured. If several diseases appear in the same sentence, each one will be its own result. The user can then focus on the captured entities, and export the entire query result to a CSV file, in which each row contains the sentence, its source, and the captured variables. In the current examples, the result will be a list of disease names that co-occur with "fatal" and "asymptomatic". We can also issue a query such as

'chem:e=SIMPLE_CHEMICAL d:e=DISEASE'

to get a list of chemical-disease co-occurrences. Using additional terms, we can narrow down to co-occurrences with specific words, and by using contextual restrictions (§4.3) we can focus on co-occurrences in specific papers or domains.

Expansions. Finally, for captured terms we also support *linguistic expansions*. After the term is matched, we can expand it to a larger linguistic environment based on the underlying syntactic sentence representation. An expansion is expressed by prefixing a term with angle brackets ⟨⟩:

'⟨⟩inf:infection asymptomatic fatal' will capture the word "infection" under the variable "inf" and *expand* it to its surrounding noun-phrase, capturing phrases like "malarialike infection", "asymptomatic infection", "chronic infection" and "a mild subclinical infection 9".

4.2 Sequential (Token) Queries

While boolean queries allow terms to appear in any order, we sometimes care about the exact linear placements of words with respect to each other. The term-specification, capture and expansion syntax is the same as in boolean queries, but here terms must match as in the query.

'interspecies transmission' looks for the exact phrase "interspecies transmission" and 'tag=NNS transmission' looks for the word transmission immediately preceded by a plural noun. By capturing the noun ('which:tag=NNS transmission') we obtain a list of terms that includes the words "bacteria", "diseases", "nuclei" and "crossspecies".

Wildcards. sequential queries can also use wildcard symbols: * (matching any single word), '...' (0 or more words), '...2-5...' (2-5 words). The query 'interspecies kind:...1-3... transmission' looks for the words "interspecies" and "transmission" with 1 to 3 intervening words, capturing the intervening words under "kind". First results include "host-host", "zoonotic", "virus", "TSE agent", "and interclass".

Repetitions. We also allow to specify repetitions of terms. To do so, the term is enclosed in [ ] and followed by a quantifier. We support the standard list of regular expression quantifiers: *, +, ?, {n,m}. For example, 'tag=DT [tag=JJ]* [tag=NN]+'.

4.3 Contextual Restrictions

Each query can be associated with contextual restrictions, which are secondary queries that oper-

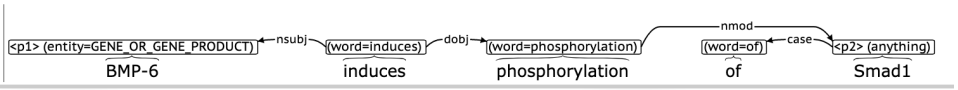


Figure 1: Query Graph of the syntactic query '⟨⟩p1:[e]BMP-6 $induces the $phosphorylation $of ⟨⟩p2:Smad1.'

ate on the same data and restrict the set of sentences that are considered for the main queries. These queries currently have the syntax of the Lucene query language.[7] Our system allows the secondary queries to condition on the paragraph the sentence appears in, and on the title, abstract, authors, publication data, publication venue and MeSH terms of the paper the sentence appears in. Additional sources of information are easy to add. For example, adding the contextual restriction '#d +title:cancer +mesh:"Age Distribution"' restricts a query results to sentences from papers which have the word "cancer" in their title and whose MeSH terms include "Age Distribution". Similarly '#d +title:/corona.*/ +year: [2015 TO 2020]' restricts queries to include sentences from papers published between 2015 and 2020 and have a word starting with *corona* in their title.

These secondary queries greatly increase the power of boolean, sequential and syntactic queries: one could look for interspecies transmissions that relate to certain diseases, or for sentence-level disease-chemical co-occurrences in papers that discuss specific sub-populations.

4.4 Example-based Syntactic Queries

Recent advances in machine learning brought with them accurate syntactic parsing, but parse-trees remain hard to use. We remedy this by employing a novel query language we introduced in (Shlain et al., 2020) which is based on the principle of query-by-example.

The query is specified by starting with a simple natural language sentence that conveys the desired syntactic structure, for example, '*BMP-6 induces the phosphorylation of Smad1*'. Then, words can be marked as anchor words (that need to match exactly) or capture nodes (that are variables). Words can also be neither anchor or capture, in which case they only support the scaffolding of the sentence. The system then translates the sentence with the captures and anchors syntax into a syntactic query graph, which is presented to the user. The user can then restrict capture nodes from "match anything" to matching specific terms (using the term specification syntax as in boolean or token queries) and can likewise relax the exact-match constraints on anchor words. Like in other query types, capture nodes can be marked for expansion. The syntactic graph is then matched against the pre-parsed and indexed corpus.

This simple markup provides a rich syntax-based query system, while alleviating the user from the need to know linguistic syntax.

For example, consider the query below, the details of which will be discussed shortly:

'⟨⟩p1:[e=GENE_OR_GENE_PRODUCT]BMP-6 $induces the $phosphorylation $of ⟨⟩p2:Smad1'

The words 'induce', 'phosphorylation' and 'of' are anchors (designated by '$'), while 'p1' and 'p2' are captures for 'BMP-6' and 'Smad1'. Both capture nodes are marked for expansion using angle braces ('⟨⟩'). Node p1 is *restricted* to match tokens with the same entity type of BMP-6 (indicated by 'e=GENE_OR...'). The query can be shortened by omitting the entity type and retaining only the entity restriction ('e'):

'⟨⟩p1:[e]BMP-6 $induces the $phosphorylation $of ⟨⟩p2:Smad1'

Here, the entity type is inferred by the system from the entity type of BMP-6.[8] The graph for the query is displayed in Figure 1. It has 5 tokens in a specific syntactic configuration determined by directed labeled edges. The 1st token must have the entity tag of 'GENE_OR...', the 2nd, 3rd, and 4th tokens must be the exact words "induces phosphorylation of", and the 5th is unconstrained.

Sentences whose syntactic graph has a subgraph that aligns to the query and adheres to the constraints will match the query. Example of matching sentences are:

- *ERK$_{p_1}$ activation induces phosphorylation of Elk-1$_{p_2}$.*
- *Thrombopoietin$_{p_1}$ activates human platelets and induces tyrosine phosphorylation of p80/85 cortactin$_{p2}$*

[7] https://lucene.apache.org/core/6_0_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html

[8] Similarly, we could specify '$[lemma]induces', resulting in the restriction 'lemma=induce' instead of 'word=induces' for the anchor.

The sentence tokens corresponding to the p1 and p2 graph nodes will be bound to variables with these names: {p1=ERK, p2=Elk-1} for the first sentence and {p1=Thrombopoietin, p2=p80/85 cortactin} for the second.

5 Example Workflow: Risk-factors

We describe a workflow which is based on using our extractive search system over a corpus of all PubMed abstracts. While the described researcher is hypothetical, the results we discuss are real.

Consider a medical researcher who is trying to compile an up to date list of the risk factors for stroke. A PubMed search for "risk factors for stroke" yields 3317 results, and reading through all results is impractical. A Google query for the same phrase brings out an info box from NHLBI[9] listing 16 common risk factors including high blood pressure, diabetes, heart disease, etc. Having a curated list which clearly outlines the risk factors is helpful, but curated lists or survey papers will often not include rare or recent research findings.

The researcher thus turns to extractive search and tries an exploratory boolean query:

'risk factor stroke'

Query

risk factor stroke

2302579	Lymphotoxin-alpha C804A polymorphism is a risk factor for stroke .
44815452	CHD was an independent risk factor for stroke , and stroke was a risk factor for CHD .
46267149	Indeed , migraine is an independent vascular risk factor of stroke , especially ischemic stroke .
49750946	Among cancer stroke patients , the potential risk factor of stroke recurrence was evaluated .

The figure shows the top results for the query and the majority of sentences retrieved indeed specify specific risk factors for stroke. This is an improvement over the PubMed results as the researcher can quickly identify the risk factors discussed without going through the different papers.

Furthermore, the top results contain risk factors like *migrane* or *C804A polymorphism* not listed in the NHLBI knowledge base. However, the full result list is lengthy and extracting all the risk factors from it manually would be tedious. Instead, the researcher notes that many of the top results are variations on the "X is a risk factor for stroke" structure. She thus continues by issuing the following syntactic query, where a capture labeled *r* is used to directly capture the risk factors:

[9] https://www.nhlbi.nih.gov/health-topics/stroke

	A	B
	r	COUNTA of r
1	Hypertension	128
2	Atrial fibrillation	45
3	AF	45
4	Diabetes	27
5	migraine	26
6	it	26
7	High blood pressure	17
8	Hyperhomocysteinemia	16
9	age	16
10	Diabetes mellitus	12
11	Arterial hypertension	11
12	smoking	9
13	Cigarette smoking	9
14	ovale	8
15	obesity	8
16	OSA	7
17	Carotid artery stenosis	7

(a) ranked risk factors for stroke

	A	B
	r	COUNTA of r
1	Hypertension	128
2	Atrial fibrillation	45
3	AF	45
4	Diabetes	27
5	migraine	26
6	Hyperhomocysteinemia	15
7	Diabetes mellitus	12
8	Arterial hypertension	10
9	obesity	8
10	OSA	7
11	Metabolic syndrome	6
12	TIA	5
13	Proteinuria	5
14	Obstructive sleep apnoea	4
15	Infection	4
16	Hyperlipidemia	4
17	Atherosclerosis	4

(b) ranked disease risk factors

Figure 2: Grouped and ranked results

'r:Diabetes is a $risk $factor for $stroke'.

Query

r:Diabetes is a $risk $factor for $stroke

954668	Low temperature might be a risk factor for ischemic stroke , and high temperature might be protective factor of ischemic stroke occurrence in Jinan , China .
2223057	Diabetes is an increased risk factor for stroke and results in increased brain damage in experimental animals and humans .
31015146	The metabolic syndrome associated with insulin resistance is also a significant risk factor for stroke .

The figure shows the top results for the query and the risk factors are indeed labeled with *r* as expected. Unfortunately, some of the captured risk factors names are not fully expanded. For example, we capture *syndrome* instead of *metabolic syndrome* and *temperature* instead of *low temperature*. Being interested in capturing the full names, the researcher adds angle brackets '⟨⟩' to expand the captured elements:

'⟨⟩r:Diabetes is a $risk $factor for $stroke'.

Query

<>r:Diabetes is a $risk$ factor for $stroke

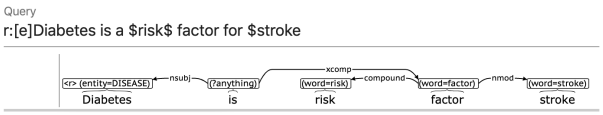
954668	Low temperature might be a risk factor for ischemic stroke , and high temperature might be protective factor of ischemic stroke occurrence in Jinan , China .
2223057	Diabetes is an increased risk factor for stroke and results in increased brain damage in experimental animals and humans .
31015146	The metabolic syndrome associated with insulin resistance is also a significant risk factor for stroke .

The full names are now captured as expected.

Now that that researcher has verified that the query yields relevant results, she clicks the download button to download the full result set.

The resulting tab separated file has 1212 rows. Each row includes a result sentence, the captured elements in it (in this case, just the risk factor), and their offsets. Using a spreadsheet to group the rows by risk factor and order the results by frequency, the researcher obtains a list of 640 unique risk factors, 114 of them appearing more than once in the data. Figure 2a lists the top results.

Reviewing the list, the researcher decides that she's not interested in general risk factors, but rather in diseases only. She modifies the query by adding an entity restriction to the 'r' capture:



As seen in the query graph, even though the researcher didn't specify the exact entity type, the query parser correctly resolved it to DISEASE. The results now include diseases like sleep apnoea and hypertension but do not include smoking, age and alcohol (see Figure 2b).

Analyzing the results, the researcher now wants to compare the risk factors in the general population to ones listed in research papers dealing with children and infants. Luckily, such papers are indexed with corresponding MeSH terms and the researcher can utilize this fact by appending '#d mesh:Child mesh:Infant -mesh:Adult' to her query. In cases where a desired MeSH term does not exist, an alternative approach is filtering the results based on words in the abstract or title. For example, appending '#d abstract:child abstract:children' to a query will ensure that the result sentences come from abstracts which contain the word *child* or the word *children*.

Happy with the results of the initial query, the researcher can further augment her list by querying for other structures which identify risk factors (e.g. "'r:Diabetes $causes $stroke'", "'$risk $factors for $stroke $include r:Diabetes'", etc.).

Importantly, once the researcher has identified one or more effective queries to extract the risk factors for stroke, the queries can easily be modified in useful ways. For example, with a small modification to our original query we can extract:

risk factors for cancer:

'r:Diabetes is a $risk $factor for $cacner'

diseases which can be caused by smoking:

'$Smoking is a $risk $factor for d:[e]stroke'.

ad-hoc KB of (risk factor, disease) tuples (for self use or as an easily queryable public resource):

'r:Diabetes is a $risk $ factor for d:[e]stroke'.

6 Example Workflow: CORD-19

The COVID-19 Open Research Dataset (Wang et al., 2020) is a collection of 45,000 research papers, including over 33,000 with full text, about COVID-19 and the coronavirus family. The corpus was released by the Allen Institute for AI and associated partners in an attempt to encourage researchers to apply recent advances in NLP to the data to generate insights.

Identifying COVID-19 Aliases Since the CORD-19 corpus includes papers about the entire Coronavirus family of viruses, it's useful to identify papers and sentences dealing specifically with COVID-19. Before converging on the acronym COVID-19 researchers have referred to the virus by many names: nCov-19, SARS-COV-ii, novel coronavirus, etc. Luckily, it's fairly easy to identify many of these aliases using a sequential pattern:

'novel coronavirus (alias:...1-2...)'

The pattern looks for the words "novel coronavirus" followed by an open parenthesis, one-or-two words which are to be captured under the 'alias' variable, and a closing parenthesis. The query retrieves 52 unique candidate aliases for COVID-19, though some of them refer to older coronaviruses such as "MERS", or non-relevant terms such as "Fig2". After ranking by frequency and validating the results, we can reuse the pattern on newly retrieved aliases to extend the list. Through this iterative process we quickly compile a list of 47 aliases. We marked all occurrences of these terms in the underlying corpus as a new entity type, COVID-19, and re-indexed the dataset with this entity information.

Exploring Drugs and Treatments. To explore drugs and treatments for COVID-19 we search the corpus for chemicals co-occuring with the COVID-19 entity using a boolean query:

'chemical:e=SIMPLE_CHEMICAL|CHEMICAL e=COVID-19'

Table 1(a) shows the top matching chemicals by frequency. While some of the substances listed like *Chloroquine* and *Remdesivir* are drugs being tested for treating COVID-19, others are only hypothesized as useful or appear in other contexts.

To guide the search toward therapeutic substances in different stages of maturity we can add indicative terms to the query. For example, the following query can be used to detect substances at the stage of clinical trials:

'chemical:e=SIMPLE_CHEMICAL|CHEMICAL e=COVID-19 l=trial|experiment', while adding 'l=suggest|hypothesize|candidate' can assist in detecting substances in ideation stage.

Table 1(b,c) shows the frequency distributions of the chemicals resulting from the two queries.

(a) Unrestricted
nucleic acid (171), chloroquine (118), nucleotide (115), NCP (87), CR3022 (47), Ksiazek (46), IgG (45), lopinavir/ritonavir (42), ECMO (40), LPV/r (35), corticosteroids (35), oxygen (32), ribavirin (31), lopinavir (31), Hydroxychloroquine (30), amino acid (30), ritonavir (27), corticosteroid (24), Sofosbuvir (22), amino acids (22), HCQ (19), glucocorticoids (19)
(b) Trial
chloroquine (29), Remdesivir (8), LPV/r (7), lopinavir (6), HCQ (6), ritonavir (4), Arbidol (4), Sofosbuvir (3), nucleotide (3), nucleic acid (3), lopinavir/ritonavir (3), CQ (3), oseltamivir (2), NCT04257656 (2), NCT04252664 (2), Meplazumab (2), Hydroxychloroquine (2), glucocorticoids(2), CEP(2)
(c) Ideation
chloroquine (6), ritonavir (5), S-RBD (4), nucleotide (4), Lopinavir (4), CR3022 (4), Ribavirin (3), nucleic acid (3), logP (3), Li (3), ledipasvir (3), IgG (3), HCQ (3), TGEV (2), teicoplanin (2), nelfinavir (2), NCP (2), HWs (2) glucocorticoids (2), ENPEP (2), ECMO (2), darunavir (2), creatinine (2), creatine (2), CQ (2), corticosteroid (2), CEP (2), ARB (2)

Table 1: Top chemicals co-occuring with the COVID-19 entity and their counts. (a) Unrestricted. (b) with Trial related terms. (c) with Ideation related terms.

While the queries are very basic and include only a few terms for each category, the difference is clearly noticeable: while the Malaria drug Chloroquine tops both lists, the antiviral drug Remdesivir which is currently tested for COVID-19 is second on the list of trial related drugs but does not appear at all as a top result for ideation related drugs.

Importantly, entity co-mention queries like the ones above rely on the availability and accuracy of underlying NER models. As we've seen in Section 5, in cases where the relevant types are not extracted by NER, syntactic queries can be used instead. For example the following query captures sentences including chemicals being used on patients (the abstract or paragraph are required to include COVID-19 related terms).

'he was $treated $with a ⟨⟩chem:treatment #d paragraph:ncov* paragraph:covid* abstract:ncov* abstract:covid*'

Treatments (via syntactic query)
ribavirin (11), oseltamivir (9), ECMO (6), convalescent plasma (4), TCM (3), LPV/r (3), three fusions of MSCs (2), supportive care (2), protective conditions (2), lopinavir/ritonavir (2), intravenous remdesivir (2), hydroxychloroquine (2), HCQ (2), glucocorticoids (2), FPV (2), effective isolation (2), chloroquine (2), caution (2), bDMARDs (2), azithromycin (2), ARBs (2), antivirals (2), ACE inhibitors (2), 500 mg chloroquine (2), masks (1)

Table 2: Top elements occurring in the syntactic "treated with X" configuration. Note that this query does not rely on NER information.

The top results by frequency are shown in Table 2. The top ranking results show many of the chemicals obtained by equivalent boolean queries[10], but interestingly, they also contain non-chemical treatments like *supportive care*, *isolation* and *masks*. This demonstrates a benefit of using entity agnostic syntactic patterns even in cases where a strong NER model exists.

7 More Examples

While the workflows discussed above pertain mainly to the medical domain, the system is optimized for the broader life science domain. Here are a sample of additional queries, showing different potential use-cases.

Which genes regulate a cell process:
'⟨⟩p1:[e]CD95 v:[l]regulates ⟨⟩p:[e]apoptosis'

Which specie is the natural host of a disease:
'⟨⟩host:[e]bat is a $natural $host of ⟨⟩disease:[e]coronavirus'

Documented LOF mutations in genes:
'$loss $of $function ⟨⟩m:[w]mutation in ⟨⟩gene:[e]PAX8'

8 Conclusion

We presented a search system that targets extracting facts from a biomed corpus and demonstrated its utility in a research and a clinical context over CORD-19 and PubMed. The system works in an Extractive Search paradigm which allows rapid information seeking practices in 3 modes: boolean, sequential and syntactic. The interactive and flexible nature of the system makes it suitable for users in different levels of sophistication.

[10] to get a more comprehensive coverage we can issue queries for other syntactic structures like '⟨⟩chem:chemical was used $in $treatment' and combine the results of the different queries.

Acknowledgements The work performed at BIU is supported by funding from the Europoean Research Council (ERC) under the Europoean Union's Horizon 2020 research and innovation programme, grant agreement No. 802774 (iEXTRACT).

References

Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 157–162.

Alan Akbik, Thilo Michael, and Christoph Boden. 2014. Exploratory relation extraction in large text corpora. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2087–2096.

Estelle Chaix, Bertrand Dubreucq, Abdelhak Fatihi, Dialekti Valsamou, Robert Bossy, Mouhamadou Ba, Louise Deléger, Pierre Zweigenbaum, Philippe Bessieres, Loic Lepiniec, et al. 2016. Overview of the regulatory network of plant seed development (seedev) task at the bionlp shared task 2016. In *Proceedings of the 4th bionlp shared task workshop*, pages 1–11.

Louise Deléger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferré, Philippe Bessieres, and Claire Nédellec. 2016. Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP shared task workshop*, pages 12–22.

Kim Jin-Dong, Nédellec Claire, Bossy Robert, and Deléger Louise, editors. 2019. *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*. Association for Computational Linguistics, Hong Kong, China.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 workshop companion volume for shared task*, pages 1–9.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011. Overview of genia event task in bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 7–15. Association for Computational Linguistics.

Svetlana Kiritchenko, Berry de Bruijn, Simona Carini, Joel Martin, and Ida Sim. 2010. Exact: Automatic extraction of clinical trial characteristics from journal publications. *BMC medical informatics and decision making*, 10:56.

Iain J Marshall, Joël Kuiper, and Byron C Wallace. 2015. RobotReviewer: evaluation of a system for automatically assessing bias in clinical trials. *Journal of the American Medical Informatics Association*, 23(1):193–201.

J.G. Mork, Antonio Jimeno-Yepes, and Alan Aronson. 2013. The nlm medical text indexer system for indexing biomedical literature. *CEUR Workshop Proceedings*, 1094.

Bernd Müller, Christoph Poley, Jana Pössel, Alexandra Hagelstein, and Thomas Gübitz. 2017. LIVIVO : the vertical search engine for life sciences. *Datenbank-Spektrum*, pages 1–6.

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

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics.

Micah Shlain, Hillel Taub-Tabib, Shoval Sadde, and Yoav Goldberg. 2020. Syntactic search by example. In *Proceedings of ACL 2020, System Demonstrations*.

Axel J Soto, Piotr Przybyła, and Sophia Ananiadou. 2018. Thalia: semantic search engine for biomedical abstracts. *Bioinformatics*, 35(10):1799–1801.

Aryeh Tiktinsky, Yoav Goldberg, and Reut Tsarfaty. 2020. pybart: Evidence-based syntactic transformations for ie. In *Proceedings of ACL 2020, System Demonstrations*.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Dane Bell. 2020. Odinson: A fast rule-based information extraction framework. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, France. European Language Resources Association (ELRA).

Patrick Verga, Emma Strubell, Ofer Shai, and Andrew McCallum. 2017. Attending to all mention pairs for full abstract biological relation extraction. *arXiv preprint arXiv:1710.08312*.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Michael Kinney, Ziyang Liu, William. Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. Cord-19: The covid-19 open research dataset. *ArXiv*, abs/2004.10706.

Lucy Lu Wang, Oyvind Tafjord, Sarthak Jain, Arman Cohan, Sam Skjonsberg, Carissa Schoenick, Nick Botner, and Waleed Ammar. 2019. Extracting evidence of supplement-drug interactions from literature. *arXiv preprint arXiv:1909.08135*.