ENTITY-RELATION SEARCH: CONTEXT PATTERN DRIVEN EXTRACTION AND INDEXING

BY

ZEQUN ZHANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor Kevin C. Chang

ABSTRACT

Our research focuses on searching relations between entities with context constraints. In particular, we are interested in efficiently searching for the relations among medical entities (e.g. diseases, chemicals, species, genes, or mutations) in a professional medical corpus. Existing relation extraction systems, like *OpenIE*, are able to extract some relations between entities. However, its results are inseparable in terms of extraction contexts, which prevents it from being able to search for the relations of given contexts.

To address this issue, we propose to build an entity-relation search system with an awareness of extraction contexts. In order to achieve this goal, we propose to extract and index contexts for each extracted relation. We evaluate our search model over millions of professional medical abstracts and show that our context indexing is effective to support the task of searching relations into contexts.

Note that this rich and novel system is the product of a collaborative team effort: Tianxiao Zhang, Jiarui Xu and Varun Berry, and supervised by Professor Kevin Chang. While we separately document our individual contributions, we intentionally share some parts of our thesis to improve the readability of our overall system design. This thesis mainly focuses on the design of our context extraction and indexing method.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1. Motivation and Challenges

Relation extraction systems aim to help users find relations from a large text corpus. Relations outputted from such systems are a set of relation phrases extracted from a mixture of contexts, which are matched by a strict extraction pattern. For example, such extraction system may find the relation between "Bill Gates" and "Microsoft" to be "founded".

Also, here are some scenarios where users are looking for relations with constraints. For example, a medical researcher may look for the relation between a specific type of disease and a gene, but she is interested in this subject with the limitation that it only pertains to mice. Or, a doctor wants to know the relation between a disease and a chemical, but only as it relates to male adult patients. While such query needs are important and usual, it is not possibly resolved by existing relation extraction systems due to their inability to differentiate contexts from which relations were extracted. This limitation prevents such systems from being able to provide context-specific results, for contexts like "mice" or "male adult patients". Moreover, the usage of a single strict pattern causes such extraction systems to overlook many extractable relations.

To resolve such queries, a system needs to preserve context information for extracted relations, and efficiently use them while resolving user queries. Conventional relation extraction systems extract relations from the corpus and discard its context information. Thus such systems are not able to recover context information when a user query is inputted. We identified two challenges for solving this problem: 1) how to extract context information for relations, and 2) how to efficiently index context information to support online search. For theses two challenges, we will discuss our solutions in Section 3 and Section 5.

1.2. Entity-Relation Search: Problem Definition

One of the limitations of existing relation extraction systems is that when users query the relationship between a pair of entities, the query result is always a combination of relations from a confused mixture of contexts, regardless of the level of interest of each context that it includes. When users are interested in relations only from a limited set of contexts, it is very difficult for the system to detect what relations are within the search scope due to the infeasibility to recover relation contexts. This is because information regarding the contexts from which relations are extracted was lost during the process of extraction. In order to find context-specific relations, users will have to navigate through all of the snippets for a relation phrase to find whether any snippet contains the relation phrase in the preferred set of contexts. After seeing enough snippets, users still need to aggregate all of the information that they saw in order to form an overall impression of relatedness for the query entities in the limited set of interested contexts.

We summarize our problem in Figure 1.1. First, for input, as queries, our relation search system let user search for relation phrases by specifying subject and object, both as entities, and a list of optional context constraints in the form of entities and keywords, which indicate user's intention of where the relations should be found. By design, the relation search is essentially search relations by context over the document collection. Context constraints intend to shape the search space within which the desired relations occur. For example, users could query the relationship between "diabetes" and "insulin" and specify the context constraint to be "mice", suggesting that only the relations with "mice" involved would be interesting.

Entity-Relation Search Query.

- **Given**: Entity collection $\mathcal{E} = \{E_1, \dots, E_N\}$ and Relation Phrase collection $\mathcal{R} = \{rp_1, \dots, rp_M\}$, over Document collection $\mathcal{D} = \{d_1, \dots, d_n\}$.
- **Input**: Query $q(< E_1, E_2 >) = E_1, E_2, CE_1, \dots, CE_m, k_1, \dots, k_l$, where entity $E_i \in \mathcal{E}$, context entity $CE_j \in \mathcal{E}$, and k is keyword.

  Output: Ranked list of $t =< rp >$, where $rp \in \mathcal{R}$, sorted by $Score(q(t))$, the query score of t.

Figure 1.1 The entity-relation search problem

Second, for output, the result is a ranked list of human-readable relation phrases. A relation phrase will be ranked higher, if it matches the query better. We denote the measure of how well t matches the query q as $Score(q(t))$, which should capture how rp describes the relationship for the pair of query entities, in the specified search context.

We emphasize that, since the scoring function determines the ranking of relation phrases, it is the central function of our relation search system. Thus, the objective of the relation search is to find from the space of $\mathcal{R}$, the matching relation phrases in ranked order by how well they match query q (i.e., how well the relation phrase captures the relationship between the given pair of query entities under the preferred contexts). As the focus of this paper, we will discuss how we developed this scoring function in the Section 3.

Relation Phrase	Relation phrase is a verb phrase that denotes a binary relation in a sentence
Context Vector	A text window convers query entity pairs and relation phrase candidates; will be sentences in our case
Context Entity	Entities co-occur with query entity pairs working as the context
Context Pattern	A pattern captures all context information needed for ranking. Foe example, (E VP E) means there is an <Entity, Verb Phrase, Entity> subsequence in the sentence.

Table 1.1: Related terminologies and definitions used in this thesis

1.3. System Overview

Followed by our problem definition, we determine a relation phrase based web search system. In this system, we divide it into three main components: extraction model, indexing model and ranking model. In the extraction section, we state our POS-Tagging sequence based approach for verb phrase extraction, pattern extraction and build inverted indexes for all entities and keywords. An overview of our entire system's architecture may be found in Figure1.2.

Sequentially, during offline, we will firstly do relation and context pattern extraction. After that, we proposed an efficient indexing model to index all relation phrases, patterns and context vectors. This indexing makes it possible for our system to efficiently recover context information for relations during online processing – and this is the focus of this thesis.

Thus, when a query is coming, we can through Query Parser to extract its' additional information such as corresponding entity types. Then, through extracting related context vectors via indexing and calculate ranking score for each relation candidate by combining precomputed span model and context pattern weights, our system can return a ranked list of relation phrases efficiently and effectively.

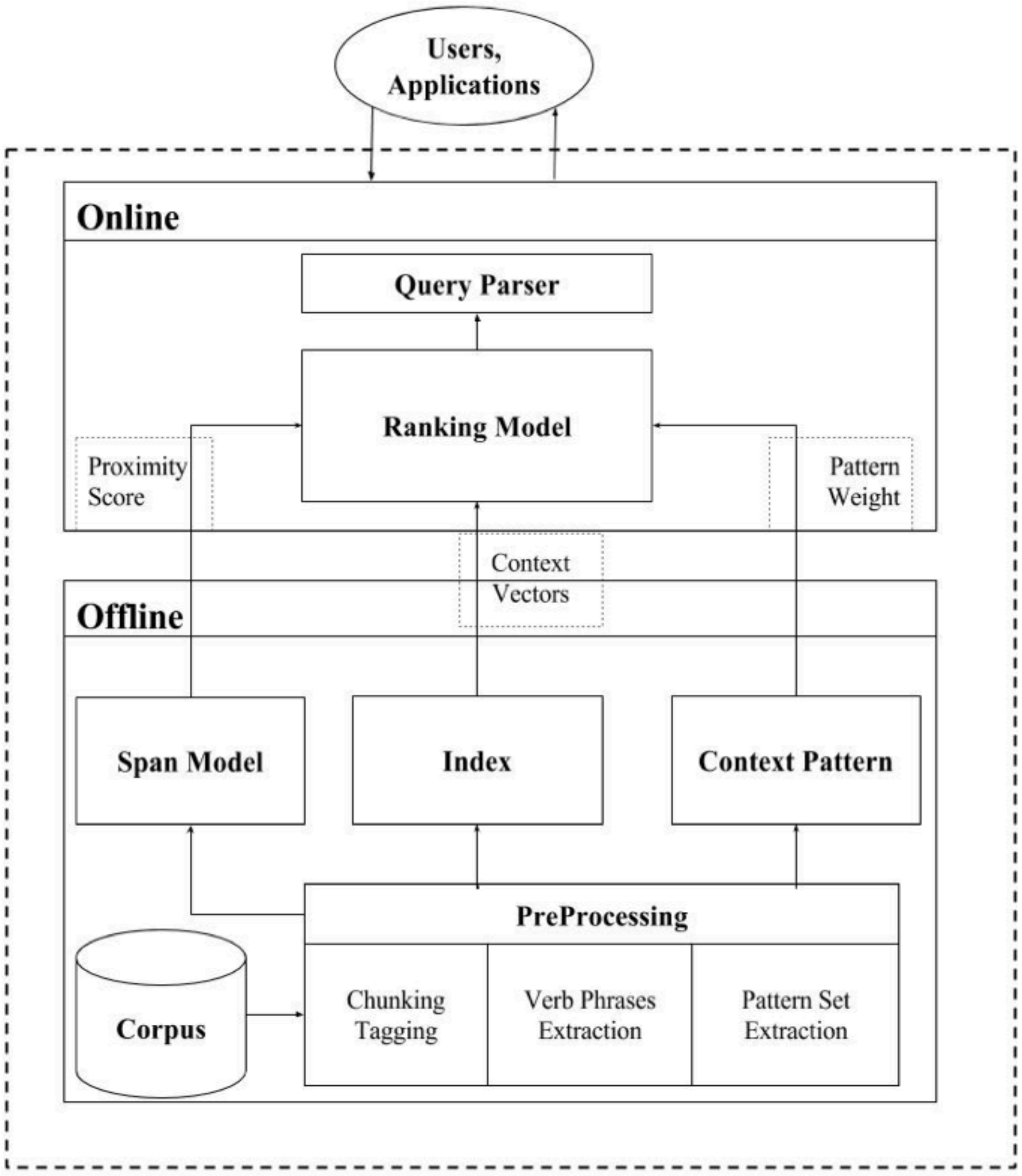


Figure 1.2: Entity Relation Search System Architecture Graph

Moreover, since this entire system is the product of a group project, the division of work we note as follows: Jiarui Xu works on the empirical studies on data insights; Varun Berry mostly focuses on the relation phrase clustering over context study; Tianxiao Zhang works on the ranking model and I work on indexing and extraction models. Also, Tianxiao and I implement the online prototype together.

1.4. My Focus: Context Extraction and Indexing

To support searching relations into contexts in real-time, a system need to recover context information for extracted relations during online processing. Conventional relation extraction systems discard context information while extracting relation tuples, which makes them incapable of doing this task. Our system, on the other hand, extract and index contexts for relations, therefore make it possible to recover context information for online ranking.

The overall *Entity-Relation Search* system's documentation is organized as follows: my thesis concentrates on the first steps of this system: extraction and indexing of context information for relation search. Remaining parts of the system, including the online ranking model of relations, are addressed in Tianxiao's thesis.

While context refers to the surroundings, in particular, we are mostly interested in entities that appear in the nearby area of a relation tuple. We claim that information from contexts is very useful to characterize relations. For example, in a medical paper, the entity "mice" may appear in the context of a relation tuple of "diabetes" and "insulin", suggesting that the relation described between disease and chemical in this paper is possibly occurred on "mice". Our extraction model is designed to extract contextual entities together with relation tuples.

Recover contexts information efficiently during online processing is difficult. To solve this task, we propose to design a special indexing structure to index all the extracted entities in the contexts with the relation tuples. Thus when we analyze relation queries online, we will be able to recover context information for relation tuples efficiently right through reading our index.

1.5. Contributions of The Thesis

We summarize contributions of the entire project as follows:

1. We discover and develop a pattern-driven based ranking model that supports search by context.
2. We introduce the novel ideas of entity modifier, context entity, and a systematic way to extract context patterns.
3. We conduct massive fundamental experiments on properties and distributions of verb phrases/patterns on a professional medical corpus.
4. We implement an online prototype on the *PubMed* corpus, and it outperforms the most popular recent work (*OpenIE*) effectively.

Individually, my contribution in this project are as follows:

1. I design and develop context extraction and indexing models that supports searching relations into contexts.
2. I propose to use context patterns to find potential relations that could be overlooked by conventional relation extraction systems.
3. I study two ways of relation phrase clustering and compare their results.

CHAPTER 2
RELATED WORK

There are two fields of study that are related to our entity relation search problem: including medical entity relation mining and entity-related search system. In the medical text-mining domain, there exists some prior work on the relationship among medical entities shown in the knowledge databases [1,2]. The most popular one is the *Comparative Toxicgenomics Database* (*CTD*) whose data, includes relations between Chemical-Gene, Chemical-Disease and Gene-Disease. Unlike the specific and predefined relations between chemical and gene by the medical professionals, the relations between disease and chemical is very general and there are only a few relation types such as “therapeutic” and “mechanism” instead of phrases. Another similar medical knowledge base is The Pharmacogenomics Knowledgebase (*PharmGKB*) which focuses on the relationship between human genetic variation and drug. Besides the issue of using predefined relation types, all of these studies need professional bio-curators to manually curate results from the scientific literature (PubMed) which is extremely time intensive, and it is very difficult to cover newly discovered medical knowledge.

In terms of the searching system side, the earliest research that proposed an entity-related search, instead of traditional link-based search engine, is the *EntityRank* [3,4]. The idea of returning a ranked entity list makes the assessment of unstructured a data-rich web more efficient and useful.

Recent researches already proposed solutions on the relation phrase automatically extraction task from unstructured corpus, in [5,6,7,8]. The one closest to our method is Open Information Extraction (*OpenIE*) [7,9]. In its process, it extracts and indexes the <subject,

relation phrase, object> tuples offline and return relations that are ordered by occurrence frequency as relation query results. Although this approach is empirically effective, it fails to allow users to search into contexts.

Considering the extraction and indexing parts that we are focusing on in this thesis, one related work is *OpenIE*. While both our system and *OpenIE* need to extract and index relation phrases, our approaches are quite different. *OpenIE* uses a strict pattern to extract relation tuples (i.e. <subject, relation phrase, object>), and index all the extracted tuples offline. In contrast, our *entity-relation search* system uses a set of more diversified patterns to capture potential relation tuples. This enables us to find relations that *OpenIE* would overlook. Moreover, instead of indexing fixed relation tuples and output them directly as relation query results, our system chooses to store entities and relation phrases separately, and only combine them into relation tuples during the online query period. This design makes it possible for our system to output context-specific relation search results that match user quires. Our indexing model is also related to *EntityRank*'s, the difference is that in our system we expand the inverted index for entities to also include context information.

CHAPTER 3
EXTRACTION MODELS

In this section we elaborate on the verb phrases extraction module and the context pattern set extraction module, as refer to Figure 1.2, that we designed for the first step of our entire system, which is the extraction task. Specifically, we group this section following the implementation order of each subtask. Note that context pattern scoring is addressed in Tianxiao's thesis.

3.1 Relation Phrase Extraction

The verb-based phrase has served as the role of a *predicate* in the conventional relation extraction system. There are two benefits of using verb phrases to describe relation. Firstly, verb-based phrases are human readable and can be easily understood; thus, they can be directly presented to users as query results. Secondly, verb-based phrases naturally exist in the original corpus and can be extracted by a certain set of POS patterns. Consequently, we utilize verb phrases to represent relation phrases. For this purpose, we adopt the same definition and extraction techniques as used in [9].

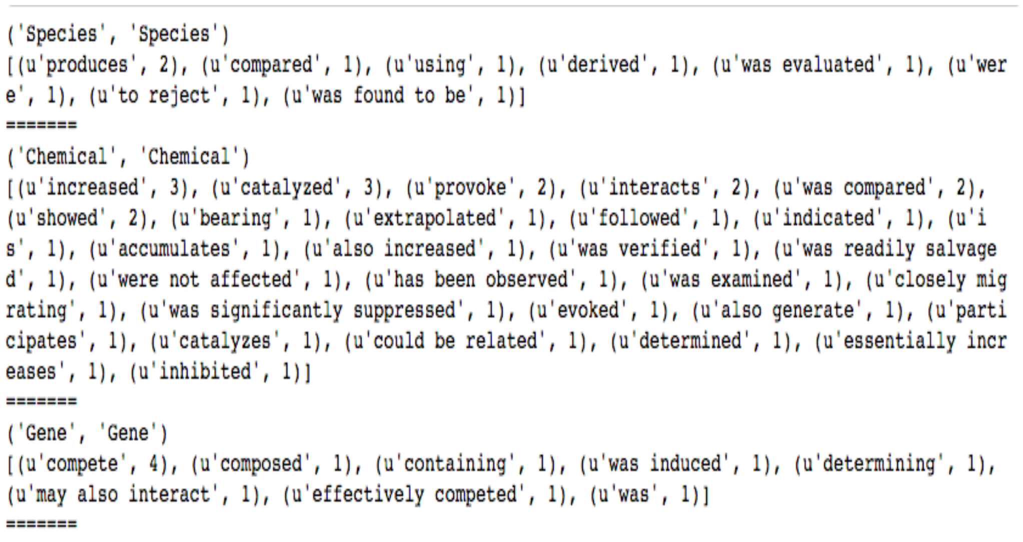
Extracted relation phrases are required to match the POS patterns shown in Figure 3.1. The patterns are designed to eliminate incoherent or uninformative extractions. The patterns require that relation phrases be a single verb (e.g. found), a verb followed by a preposition (e.g. lives in), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (e.g. is treatment of). The extractor makes a one-time scan over each sentence to extract all of the verb phrases. If two verb phrases are adjacent to each other, we merge them into a single phrase. This refinement enables the extraction of phrases that contain multiple verbs (e.g. can be treated with).

Verb Phrase Extraction Pattern (VP) = V | V+P | V+N*+P

V = Verb
N = Noun | Adjective | Adverb | Pronoun | Determiner
P = Preposition | Particle

Figure 3.1 POS-Tagging patterns used to extract relation phrases.

As constrained by the design of the extraction patterns, all extracted relation phrases must be a contiguous span of words in the original sentence. For the scope of our relation search engine, all of the verb phrases are extracted and indexed offline to enable fast online processing. A big difference between our work and conventional relation extraction systems is that all pieces of extracted relation phrases in our system are indexed independently with given contexts. Figure 3.2 shows some of the relation phrases that we extracted from the medical domain.



```
('Species', 'Species')
[(u'produces', 2), (u'compared', 1), (u'using', 1), (u'derived', 1), (u'was evaluated', 1), (u'wer
e', 1), (u'to reject', 1), (u'was found to be', 1)]
=======
('Chemical', 'Chemical')
[(u'increased', 3), (u'catalyzed', 3), (u'provoke', 2), (u'interacts', 2), (u'was compared', 2),
(u'showed', 2), (u'bearing', 1), (u'extrapolated', 1), (u'followed', 1), (u'indicated', 1), (u'i
s', 1), (u'accumulates', 1), (u'also increased', 1), (u'was verified', 1), (u'was readily salvage
d', 1), (u'were not affected', 1), (u'has been observed', 1), (u'was examined', 1), (u'closely mig
rating', 1), (u'was significantly suppressed', 1), (u'evoked', 1), (u'also generate', 1), (u'parti
cipates', 1), (u'catalyzes', 1), (u'could be related', 1), (u'determined', 1), (u'essentially incr
eases', 1), (u'inhibited', 1)]
=======
('Gene', 'Gene')
[(u'compete', 4), (u'composed', 1), (u'containing', 1), (u'was induced', 1), (u'determining', 1),
(u'may also interact', 1), (u'effectively competed', 1), (u'was', 1)]
=======
```

Figure 3.2 Several verb phrases samples grouped by entity type pair

3.2 Context Pattern Extraction

In our system, context patterns are used to capture potential relations, as we observe relations often appears in certain patterns. There are three types of context pattern components, including entity, verb phrase and entity modifier (examples in Table 3.1).

Example	Context Pattern	Context Vector
1	Entity Modifier – Entity – Verb Phrase – Entity Modifier – Entity	Nocturnal(EM) asthma(E) uncontrolled inhaled(VP) corticosteroids(EM) theophylline(E).
2	Entity Modifier – Entity – Entity Modifier – Verb Phrase – Entity - Entity Modifier	Serum(EM) theophylline(E) concentrations(EM) determined(VP) theophylline(E) dosage(EM).

Table 3.1: Two context vector with its context patterns

Entity modifier is introduced to specify a sub-level of the entity or to describe a relation under a certain condition. Note that by adding entity modifiers, our system is able to not only further distinguish entity relations from general types (e.g. the only pattern in the *OpenIE*: *E VP E*) to find more specific ones, and it assists users to better understand the relation phrases. For example, an entity modifier could be used to explain the occurrence of opposite relation phrases for the same query (e.g. aspirin can treat a headache while aspirin with alcohol can cause a headache).

According to our observation from corpus, we intentionally limit the entity modifier to be either an adjective or noun (not entity) that is directly before an entity or noun (not entity) and directly after an entity; Even though an entity modifier and entity is located close to each other, we allow for a jump of words between verb phrase and entities.

We explain the detailed usage and scoring of context patterns in Tianxiao's thesis.

CHAPTER 4
RELATION CLUSTERING

Similar to other medical text-mining problems, we suffer from the sparsity problem. Most entities co-occur only a few times in the *PubMed* corpus and there are often diverse ways to describe the same meaning relation between an entity pair. To conquer this issue and take the leverage of the redundancy of the corpus, we decided to cluster synonymous relation phrases.

For relation vector based clustering, and to follow the traditional principle for such a task, we ran a k-means clustering method on the relation phrases, which is represented by relation vectors (Figure 4.1). The relation vector is basically a bag-of-words model, which contains TF-IDF values, multiplying the occurrence frequency for each term. In addition, we observe that the meaning of the relation phrase becomes ambiguous without considering entity type information. For example, “prevent” and “treat” are of similar relations to “chemical” and “disease”, but they should be view differently to “gene” and “chemical”. Thus, we added the query entity pair’s type information to the relation vector. Finally, we take account of relation phrase polarity information to cluster similar the semantic sense of relation phrases. We chose the mass center vector of each cluster to represent the group, in order to leverage the result.

Relation Vector on K-Means Clustering Formula:
Mass of cluster j: $m_j = \sum_{i=1}^{N_j} \lvert CV_i \rvert$
Mass center vector of cluster j: $x_m = \frac{\sum_{i=1}^{N_j} \lvert CV_i \rvert CV_{im}}{m_j}$

Figure 4.1 Relation vector definition

Another more sophisticated solution is to cluster based on the relation phrases' contexts instead of its own content. For this approach, we suggest to cluster on experiment context vectors first to acquire word group information. Then, for each relation phrase, we remove it from the context vector and preform chunking to retrieve many consecutive words. For each substring, we use its word's group information for the vector representation. Considering the level of efficiency for the large-scale corpus, it is recommended to use *MinHash* [11] and *LSH* on those sets of vectors to compute the *Jaccard* similarity among those relation phrases. To better explain this idea, we show a simple example in Figure 4.2 to elaborate the entire process. Theoretically, the first approach groups relation phrases (if they share common/similar terms in the phrases), and the second approach defines similar relation phrases by evaluating whether they coexist in a similar context.

Relation Context on *MinHash* + *Locality Sensitive Hashing* example:

Input:

Context vector 1: E1 E2 RP1 EM1 E3
Context vector 2: E2 E4 RP2 E5 EM1

Step 1: *Word2vec* based word clustering:

Assume result as follows:
Group 1 – E1 E3 E4
Group 2 – E2 EM1 E5

Step 2: Relation Phrase Set Representation (Chunking by three-word shingles):

RP1 = (122,221) RP2 = (212,122)

Step 3: *MinHash* and LSH to compute *Jaccard* similarity between RP1 and RP2:

Output: 1/3

Figure 4.2 An example of context based relation phrase clustering

CHAPTER 5
INDEXING

This section describes the index module in Figure 1.2. In our system, context indexing is used to preserve context information for online relation search. We stress that it can be easily implemented based on existing search engine infrastructure for interactive online search.

We now discuss a possible implementation for our context indexing. To begin with, we assume that a document collection has been transformed into an entity collection, by using entity extraction techniques.

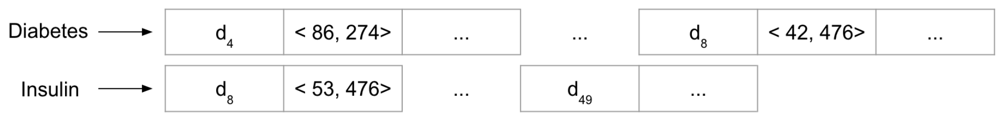


Figure 5.1: Indexing example

We use the standard inverted index for indexing keywords. To index an entity with contexts, our system will produce a list containing all the information regarding its context. To be specific, the list records for each entity, the position of the extraction in the documents (e.g. position 42 at document 8), and the context vector ID (e.g. 476). Each context vector ID maps to a context vector that stores a list of extracted verb phrases and entity modifiers. If two entities co-occur within a small text window, they might share the same context vector, thus the same context vector ID. As shown in Figure 5.1, all the context information of occurrences of an entity is stored in a list that is ordered by document number.

Now we examine how to efficiently perform *entity-relation search* upon such an index. The *entity-relation search* algorithm is demonstrated in Algorithm 5.1. Let us work through this algorithm for the relation query “diabetes insulin” upon the index in Figure 5.1. First, we load the inverted index for “diabetes” and “insulin” (line 0). Then we iterate the two lists in parallel, checking for any intersecting documents in line 1. In this example, the algorithm will report the first intersecting document to be d_8. Then, the algorithm will further check if a tuple, forms by the pair of entities “diabetes” and “insulin” and any verb phrase (e.g. “treated with”) recovered from its context vectors, satisfies any context pattern (e.g. “*E VP E*”) in our pre-defined pattern set. If a matching tuple is found, we will then calculate the matching score for it in line 4. Finally, after we initiated all possible tuples, we aggregate the scores for each tuple in line 6 and output it as its ranking score in line 7.

We note that the core of our *Entity-Relation Search* algorithm (lines 1-4) is essentially performing sort-merge-join over parallel ordered lists. By design, our algorithm can be run very efficiently. In addition, since this sort-merge-join works on a document basis, it can easily be fully parallelized, by partitioning the entire corpus into sub-corpuses. This parallelism provides superior possibilities to support real-time large-scale entity-relation search.

The Relation Search Algorithm:

Given: $L(E_i),\ L(K_j)$: $inverted\ lists\ for\ all\ the\ entities\ and\ keywords;$
S_P: $context\ pattern\ set.$
Input: $q = E_1, E_2, CE_1, \dots, CE_m, k_1, \dots, k_l$: entities, context entities and keywords.

```
0:  Load inverted lists: $L(E_1), L(E_2), L(CE_1), \dots, L(CE_m), L(K_1), \dots, L(K_l)$;
    $/*\ intersecting\ lists\ by\ context\ number$
1:  **For** each doc $d$ in the intersection of all lists:
2:      Use context pattern $p \in S_P$ to initiate tuples;  $/*\ matching$
3:      **For** each instantiated tuple $t$:
4:          Calculate $p(q(t)|d)$;
5:  **For** each tuple $t$ initiated in the whole process:
6:      calculate $P(q(t)|D) = \sum_d P(q(t)|d) * P(d)$
7:      output $Score(q(t)) = P(q(t)|D)$
```

Algorithm 5.1: The relation search algorithm

CHAPTER 6
EXPERIMENTAL RESULTS

We evaluate our system on the *PubMed* professional medical abstracts. We take advantage of the entity and type information obtained from *PubTator* [13], an entity detection and extraction tool in the PubMed data set. Specifically, it provides five entity types and its occurrences in the PubMed – Disease, Chemical, Gene, Mutation and Species. We will present our implementation results in the following order: Sections 6.1 and 6.2 will present some empirical studies about verb phrases cover rate and verb phrases clustering. Sections 6.3 and 6.4 reveal the performance and interface of our online prototype. Note that the size of data used in different experiments could be various and the exact setting will be discussed in each separate section.

6.1 Experiments on Verb Phrases Cover Rates

We want to figure out whether a relatively small finite set of relation phrases can "cover" majority entities' relationships. We define a mathematical formula to capture such impressions in Figure 6.1. Intuitively, we computed the expectation of a relation set covering the entire corpus. Not surprisingly, we conclude that a small and finite relation candidate set could approximately cover most of the relations from around 20 million abstracts, as shown in Figure 6.2. These two studies guide and support us to extract relation phrase candidate sets in a more reasonable way.

$$Cover(RPS,EPS) = \sum_{i=1}^{N} P(EP_i) f(EP_i,RPS) = \sum_{i=1}^{N} \frac{Count(EP_i)}{|D|} * \frac{r(EP_i) \cap RPS}{|r(EP_i)|}$$

EP: Entity Pair <E1,E2>

EPS: A set of entity pairs

RPS: A relation phrase set order by frequency

N: Number of entity pairs

$P(EP_i)$: prior probability of a query on EP_i

$f(EP_i, RPS)$: confidence level of RPS explaining EP_i

$r(EP_i)$: A set of relation phrases between EP_i

Figure 6.1: Definition of cover rate between a set of entity pairs and a set of verb phrases

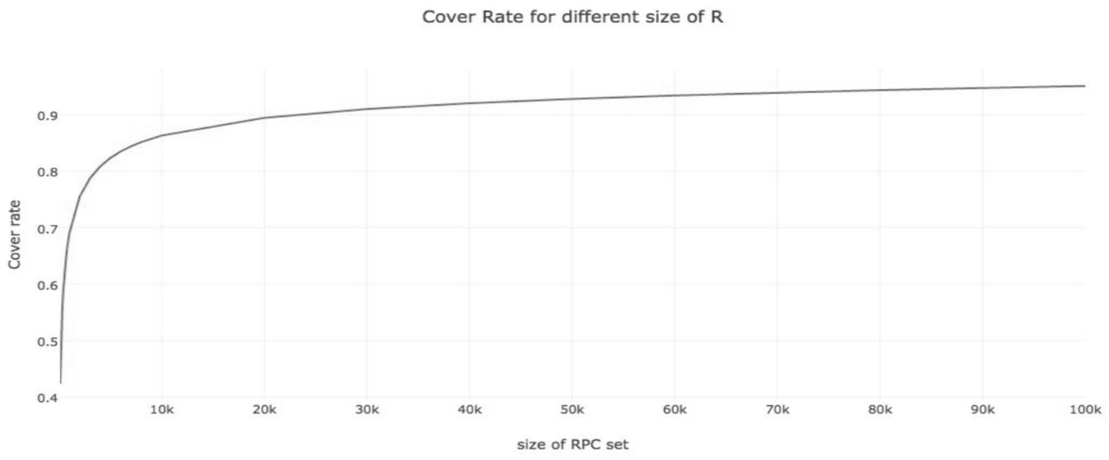


Figure 6.2: Relation between the size of the relation phrase set ordered by frequency and its cover rate

6.2 Relation Phrases Clustering

Verb phrases clustering plays an important role in our system. In particular, it helps to alleviate the problem of relation sparsity, both in the extraction process and final outputs. We now examine the the results from two of our verb clustering approaches. This is elaborated in Section 4, which addresses relation vector based clustering and context based clustering. We start with a set of relation phrases that we extracted for the entity pair "Obesity" and "Insulin".

In Table 6.1, clusters of verb phrases for "Obesity" and "Insulin" are presented in groups. In general, the second method which groups relations by its contexts produces better results in this case. Specifically, if we look at the second cluster generated by this method, those verb phrases actually refer to what the scientists do in the medical domain.

Clusters	***Relation Vector Clustering***	***Context Clustering***
Cluster 1	*associated* *compared* *appear* *account for* *develop*	*played a role in* *associated*
Cluster 2	*played a role in* *measured* *examined* *study* *investigate whether* *showed* *observed* *modulate* *improved* *help* *provide good glycaemic control in*	*compared* *measured* *examined* *study* *investigate whether*
Cluster 3	*truncated*	*appear* *showed* *truncated* *observed* *account for* *modulate*
Cluster 4		*develop* *improved* *help* *provide good glycaemic control in*

Table 6.1: Relation clusters

Our second experiment was conducted and based on the relations associated with "Diabetes" and "Obesity". Table 6.2 indicates the clustering results. Again, clusters generated by clustering relation vectors tend to have unbalanced cluster sizes, and relations are not clustered properly. In contrast, the results from context clustering captures some interesting relation phrase clusters. For example, Cluster 1 by context clustering reports that studies in the medical domain often involves a comparison of two variables. However, we note in Cluster 4, that we group “control” and “normalized” with other verb phrases in this cluster. This is probably because “control” and “normalized” appear in sentences such as, “the effect was doubled/tripled compared to the control…”. Here, the verb used to represent an increase in something is situated in the same context as words used commonly to report findings such as a 'control' group and 'normalized' results. We believe a more careful examination of contexts would help to avoid these kinds of mistakes.

Clusters	***Relation Vector Clustering***	***Context Clustering***
Cluster 1	*be a more important risk factor for*	*be a more important risk factor for* *be a more important predictor of* *found a negative correlation between* *sought an association between*
Cluster 2	*known* *influencing* *increased*	*assess the independence of* *evaluate the association of*
Cluster 3	*be a more important predictor of* *found a negative correlation between* *sought an association between* *assess the independence of* *evaluate the association of* *influenced* *reduces the development of* *be a link between* *led* *control* *depended* *rose* *doubled* *tripled* *sleep* *normalized*	*control* *depended* *rose* *doubled* *tripled* *sleep* *normalized* *known* *influencing* *increased*
Cluster 4		*influenced* *reduces the development of* *be a link between* *led*

Table 6.2: Relation clusters for “Obesity” and “Insulin”

6.3 Comparison with *OpenIE*

To evaluate the quality of our relation phrase ranking results, we compare them with the most famous relation extraction system - *OpenIE*. We observe that *OpenIE* does not index some of our medical entities, thus, their extraction result would be empty. In order to conduct a fair comparison, we adopted the *OpenIE* algorithm on our PubMed corpus.

We compare the result qualities of *Entity-Relation Search* and *OpenIE* by showing the *precision* of ranking results at different ranks for a same set of relationship queries. As shown in Table 6.3, we manually collected twenty pairs of query entities, covering some popular entities of diseases, species, chemicals and genes. We expect the ground truth relations for the set of testing quires we build to cover both obvious and obscure relationships.

As a result, Figure 6.3 shows the *precision* of the relation query results at each of the ranks for both *Entity-Relation Search* and *OpenIE*. This result is built by executing all of the relation queries listed in Table 6.3 on both of the two systems, and by manually inspecting whether each returned relation phrase holds true for its corresponding query entities. As the figure indicates, the precision of results generated by *Entity-Relation Search* generally outperforms those extracted by *OpenIE*, for the top 20 positions. Among the top five, *Entity-Relation Search* achieves 18.9% improvements in precision over *OpenIE*, which demonstrates the superior performance of its ranking model.

We also notice that, *OpenIE* performs better for the *precision* at the top-ranked result. Our analysis of this result is that, as we try to capture more relations by diversified context patterns in our *Entity-Relation Search* system, we tend to include in our results, some false positive relations that are just popular in the particular contexts. We will leave this problem for our future work.

	Query Entity Types	**Query Entities**
1	Disease - Chemical	Obesity - Glucose
2	Disease - Chemical	Asthma - Aminophylline
3	Disease - Gene	Diabetes - Insulin
4	Disease - Species	Obesity - Children
5	Disease - Species	Cancer - Children
6	Disease - Species	Breast Cancer - Children
7	Disease - Species	Cancer - Human
8	Disease - Species	Influenza- Children
9	Disease - Species	Tumor - Mice
10	Disease - Disease	Obesity - Diabetes
11	Disease - Disease	Tumor - Cancer
12	Chemical - Gene	Oxygen - BNP
13	Chemical - Species	Calcium - Children
14	Chemical - Species	Oxygen - Dog
15	Chemical - Chemical	Glucose - Serine
16	Chemical - Chemical	Cholesterol- Glucose
17	Species - Species	Human - Rats
18	Species - Gene	BNP - Patient
19	Species - Gene	Tau - Human
20	Species - Gene	PCNA - Human

Table 6.3: Test Queries

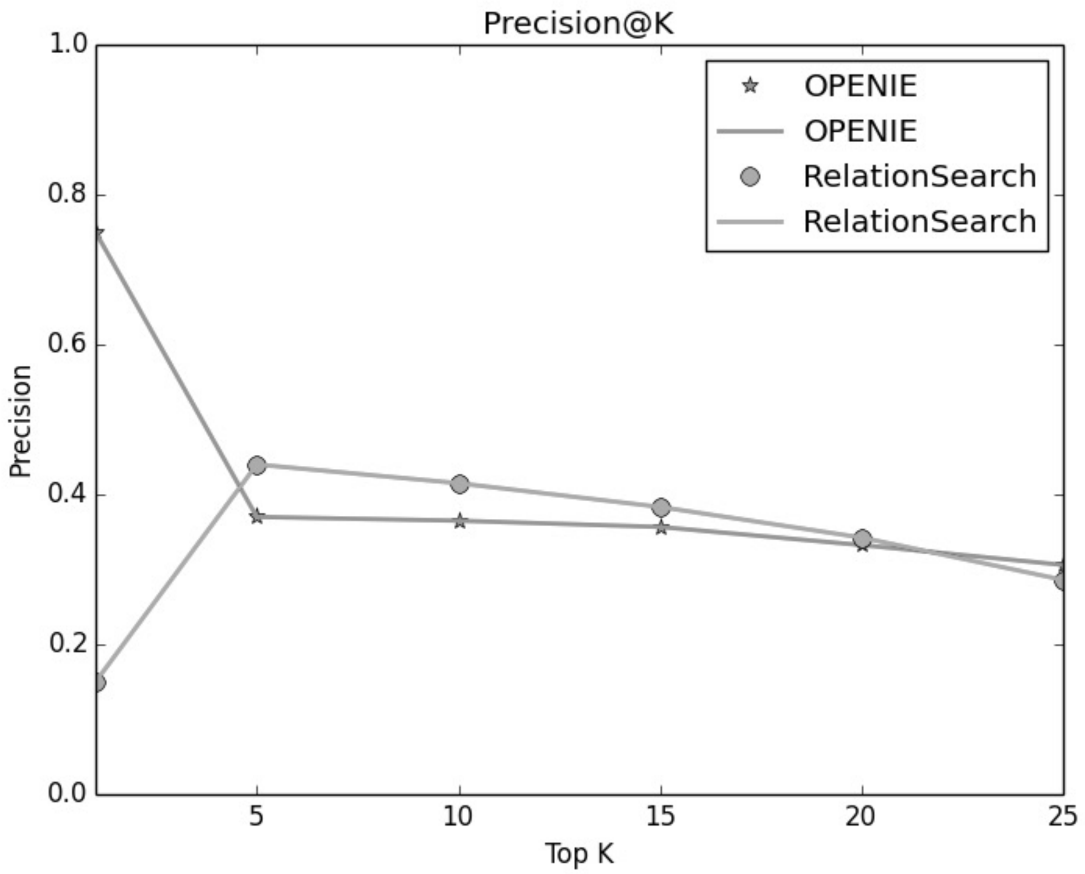


Figure 6.3: Precision at K for test queries on *OpenIE* and *Entity-Relation Search*

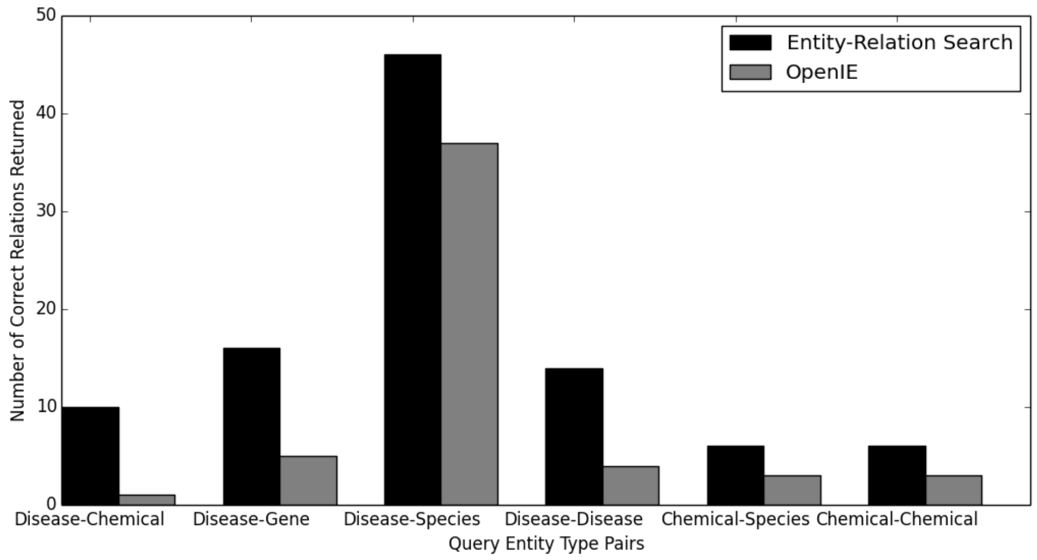


Figure 6.4: Number of correct relations discovered by *Entity-Relation Search* and *OpenIE* for each query type pair

In a corpus as large as *PubMed*, the ground truth of all relations is hard to determine. In order to compare the *recall* from both of the two systems, we manually examine the number of correct relations that each system discovers and report the results in Figure 6.4, for the 20 queries listed in Table 6.3. Note that results in Figure 6.4 is grouped by entity type pair in queries (i.e. results under "Disease-Species" refer to query 4-9 in Table 6.3). This result clearly demonstrates that the pattern-driven relation extraction method used in *Entity-Relation Search* is more capable of extracting correct relations than *OpenIE*. As we can see, *Entity-Relation Search* is able to find more correct relations for all 6 type pairs. For example, for queries containing the "Disease-Species" pair, *Entity-Relation Search* is able to find *24.3%* more correct relations than *OpenIE*. This is because the context pattern set that we use to match relation tuples is much more diversified than *OpenIE*'s strict "*E VP E*" pattern.

One significant difference between our system and *OpenIE* is that, we support searching relations between entities with context constraints. Since it is not supported *OpenIE*, we will only test it on our system. Please refer to our case studies in Section 6.4.

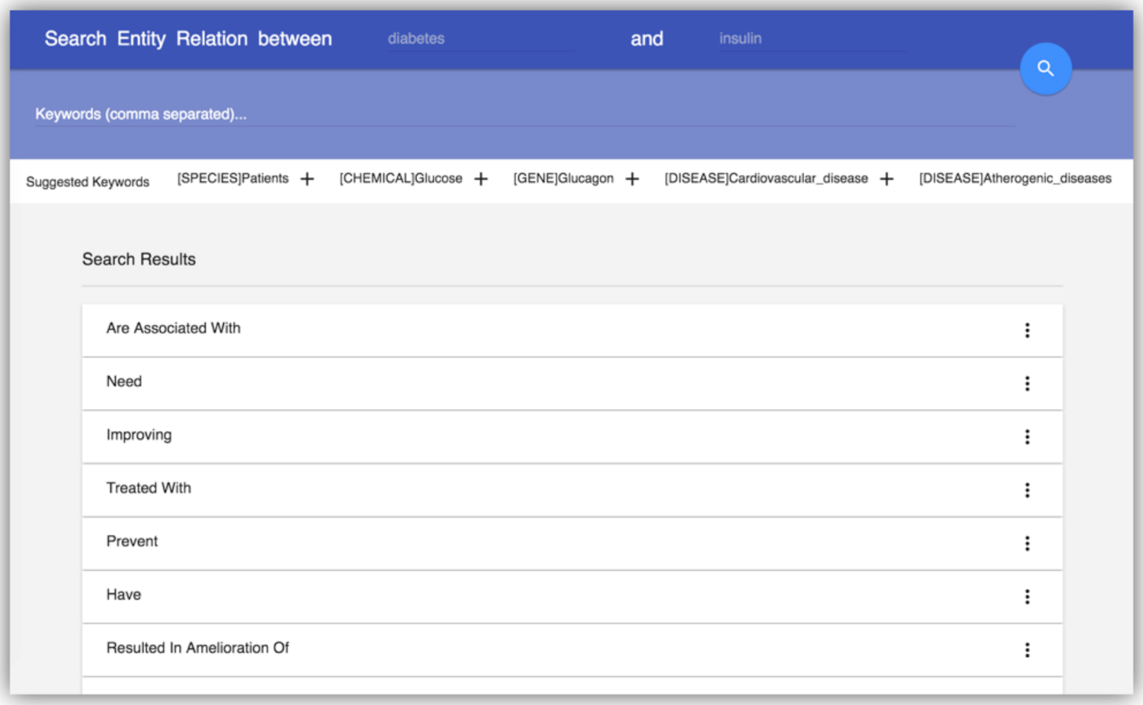
6.4 Case Study and Demo Interface

We evaluate the performance of our search system through two case studies. These studies should reflect the design of relation phrase grouping and ranking in our system.

Case Study 1:

Suppose a user wants to know the relations between "diabetes" and "insulin". She inputs these two keywords at the top of our UI and clicks on the "search" button on the right side. The search engine returns a list of relation phrases as shown in the screenshot below. These human-readable text phrases are ranked by our ranking model to best describe the relationships between the pair of query entities. The first one is “are associated with”, indicating that “diabetes” is associated with “insulin”.

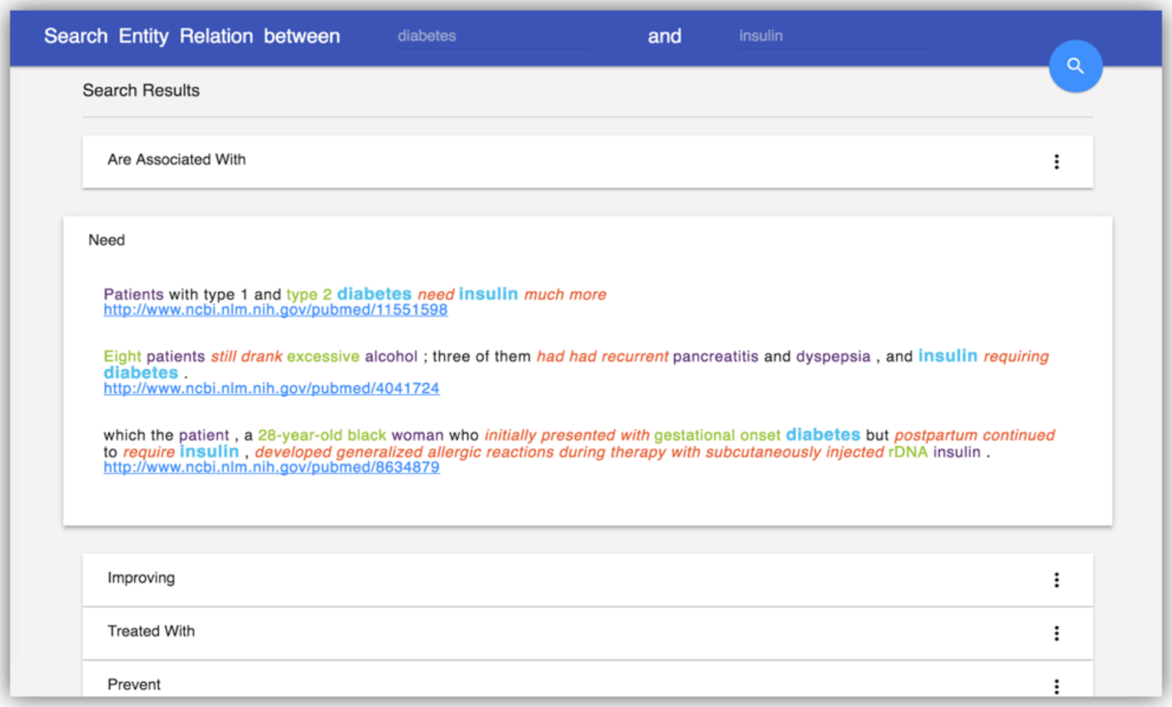
Immediately below the search boxes, we also provide a list of ranked sub-contexts represented by keywords. Users could click on “add” to add these keywords/entities to continue query context-specific relation results.


Search Entity Relation between
diabetes
and
insulin
Keywords (comma separated)...
Suggested Keywords
[SPECIES]Patients
[CHEMICAL]Glucose
[GENE]Glucagon
[DISEASE]Cardiovascular_disease
[DISEASE]Atherogenic_diseases
Search Results

Are Associated With
Need
Improving
Treated With
Prevent
Have
Resulted In Amelioration Of

Figure 6.5: Query result of the *Entity-Relation Search* system for “Diabetes” and “Insulin”

If we click on a result, for instance, the second one “Need”, we will see a list of evidences where we extract the relationship. In this view, query entities “diabetes” and “insulin” are shown in BLUE, relation phrases are in RED and entity modifiers are in GREEN. They are all part of our extraction patterns. In this example, the three relation phrases “need”, “requiring”, and “require” are clustered together under the relation phrase “Need”.


Search Entity Relation between diabetes and insulin

Search Results

Are Associated With

Need

Patients with type 1 and type 2 diabetes need insulin much more
http://www.ncbi.nlm.nih.gov/pubmed/11551598

Eight patients still drank excessive alcohol ; three of them had had recurrent pancreatitis and dyspepsia , and insulin requiring diabetes .
http://www.ncbi.nlm.nih.gov/pubmed/4041724

which the patient , a 28-year-old black woman who initially presented with gestational onset diabetes but postpartum continued to require insulin , developed generalized allergic reactions during therapy with subcutaneously injected rDNA insulin .
http://www.ncbi.nlm.nih.gov/pubmed/8634879

Improving

Treated With

Prevent

Figure 6.6: Pattern annotated snippet evidences for relation phrase “Need”

Case Study 2:

In this example we want to find the relations between entities “diabetes” and “obesity”. The results are below:

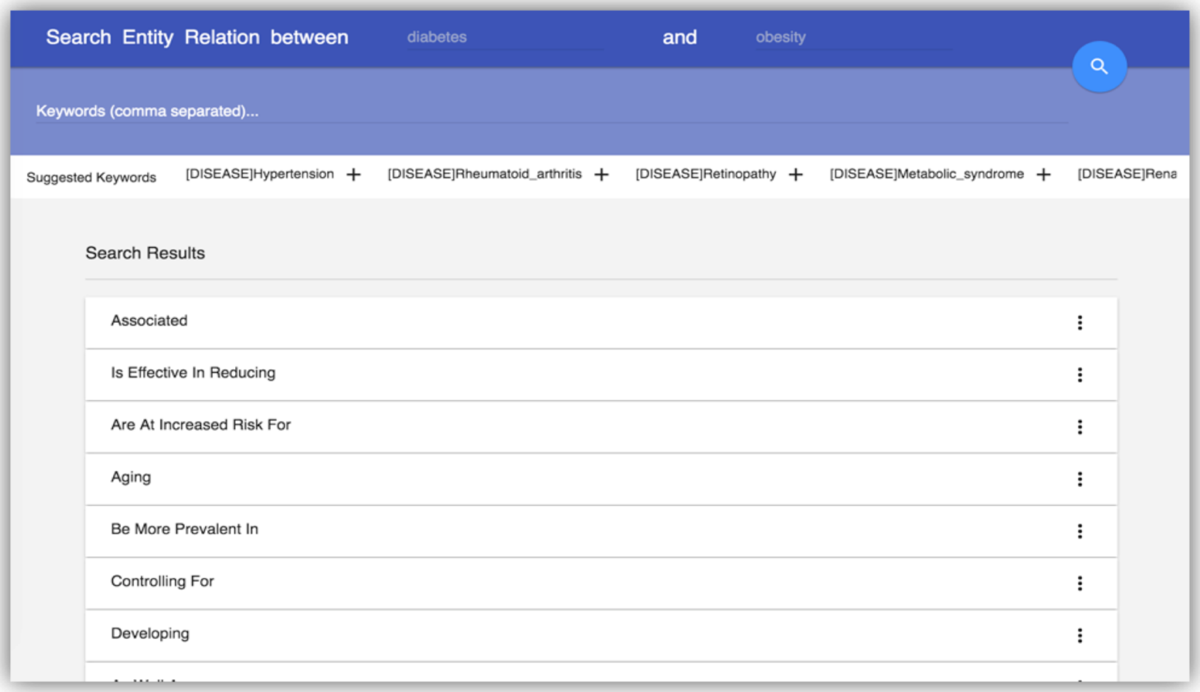
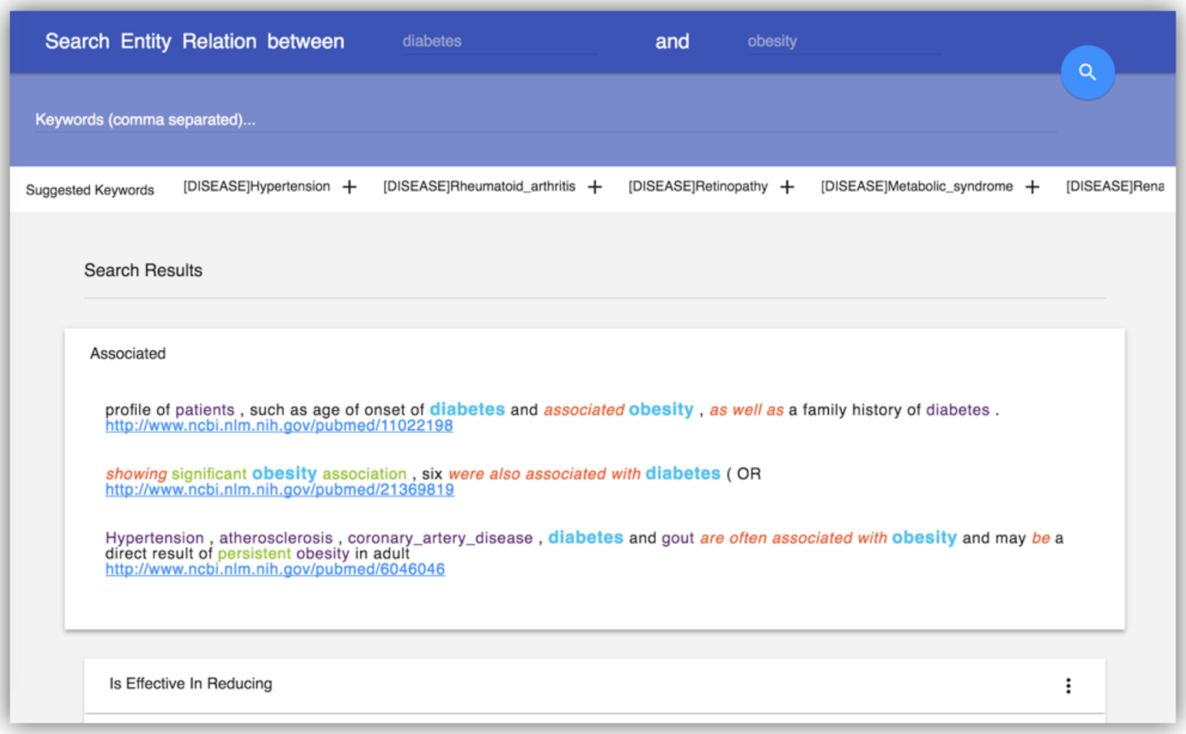


Figure 6.7: Query result of the *Entity-Relation Search* system for "Diabetes" and "Obesity"

If we click on the first result "Associated", we will expend the interface and see three text snippets (see the screenshot below). For each piece of snippet, there is a hyper-link to its original source. Entities that appeared in the context but not in the query will be presented in PURPLE. In this example, relation phrases "associated", "were also associated with" and "are often associated with" are grouped together.



Associated

profile of patients , such as age of onset of diabetes and associated obesity , as well as a family history of diabetes .
http://www.ncbi.nlm.nih.gov/pubmed/11022198

showing significant obesity association , six were also associated with diabetes (OR
http://www.ncbi.nlm.nih.gov/pubmed/21369819

Hypertension , atherosclerosis , coronary_artery_disease , diabetes and gout are often associated with obesity and may be a direct result of persistent obesity in adult
http://www.ncbi.nlm.nih.gov/pubmed/6046046

Figure 6.8: Pattern annotated snippet evidences for relation phrase “Associated”

CHAPTER 7
CONCLUSION AND FUTURE WORK

In this paper, we propose the notion of context indexing, to solve the problem of searching relations with context constraints in an unstructured text corpus. The system preserves context information for each extraction of the relation phrase and restores it for online processing. Our online prototype proves that our context indexing is able to support efficient real-time relation queries.

We identified several promising directions that our relation search system can explore in the future. First of all, we plan to build a better model to improve the precision of our relation tuple extraction. Secondly, we would like to explore other possible ways to present our search results. For example, we can present search results chronologically, to reflect changes in the time dimension. Lastly, we will excitingly extend our method to adapt to other domains, and explore new and related possibilities.

This project is a collaborative work with Tianxiao Zhang. While we separately document our individual contributions, we intentionally share some parts of our thesis to improve the readability of our overall system design.

REFERENCES

[1] Davis AP, Grondin CJ, Lennon-Hopkins K, Saraceni-Richards C, Sciaky D, King BL, Wiegers TC, Mattingly CJ. The Comparative Toxicogenomics Database's 10th year anniversary: update 2015. Nucleic Acids Res. 2014 Oct 17; pii: gku935.

[2] M. Whirl-Carrillo, E.M. McDonagh, J. M. Hebert, L. Gong, K. Sangkuhl, C.F. Thorn, R.B. Altman and T.E. Klein. "Pharmacogenomics Knowledge for Personalized Medicine" Clinical Pharmacology & Therapeutics (2012) 92(4): 414-417.

[3] Cheng, Tao, and Kevin Chen-Chuan Chang. "Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web." CIDR. Vol. 2007. 2007.

[4] Cheng, Tao, Xifeng Yan, and Kevin Chen-Chuan Chang. "EntityRank: searching entities directly and holistically." Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007.

[5] Banko, Michele, Oren Etzioni, and Turing Center. "The Tradeoffs Between Open and Traditional Relation Extraction." ACL. Vol. 8. 2008.

[6] Wu, Fei, and Daniel S. Weld. "Open information extraction using Wikipedia."Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2010.

[7] Fader, Anthony, Stephen Soderland, and Oren Etzioni. "Identifying relations for open information extraction." Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

[8] Etzioni, Oren, et al. "Open Information Extraction: The Second Generation."IJCAI. Vol. 11. 2011.

[9] Rajaraman, Anand, and Jeffrey D. Ullman. Mining of massive datasets. Vol. 1. Cambridge: Cambridge University Press, 2012.

[10] Radim Rehurek, and Sojka, Petr. "Software framework for topic modelling with large corpora." InProceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. (2010): 45-50

[11] Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.

[12] Elasticsearch. https://www.elastic.co/products/elasticsearch

[13] Wei, Chih-Hsuan, Hung-Yu Kao, and Zhiyong Lu. "PubTator: a web-based text mining tool for assisting biocuration." Nucleic acids research (2013): gkt441.