Lattice Queries for Search and Data Exploration

Boris Galitsky
Knowledge-Trail Inc. San Jose CA USA
bgalitsky@hotmail.com

Abstract

We introduce the technique of lattice querying which automatically forms the query from the set of text samples provided by a user, and produce search results by matching parse trees of this query with that of candidate answers. Lattice queries allow increase in big data exploration efficiency since they form multiple "hypotheses" concerning user intent and explore data from multiple respective angles. An importance of the lattice queries in data exploration is that only the most important keywords need to be submitted for web search, and neither a single document nor a keywords overlap delivers such the set of keywords. An open source plugin for ElasticSearch and a search request handler for SOLR are developed so that the proposed technology can be easily integrated with industrial search engines.

Introduction

Today, it is hard to overestimate the popularity of information access via search engines. Also, a vast number of distributed computing frameworks have been proposed for big data. They provide scalable storage and efficient retrieval, capable of collecting data from various sources, fast moving and fairly diverse. Modern open source big data search and exploration systems like Solr and Elasticsearch are broadly used for access and analysis of big data. However, intelligence features such as search relevance and adequate analysis, retrieval and exploration of large quantities of natural language texts are still lacking. Natural language text is still treated as a bag of words with their statistics in most industrial systems. In spite of the extensive capabilities of natural language parsing, they are still not leveraged by most search engines.

Also, frequently novice users of search engines experience difficulties formulating their queries, especially when these queries are long. It is often hard for user who is new to a domain to pick proper keywords. Even for advanced users exploring data via querying, including web queries, it is usually hard to estimate proper generality / specificity of a query being formulated. Lattice querying makes it easier for a broad range of user and data exploration tasks to formulate the query: given a few examples, it formulates the query automatically.

In this work we intend to merge the efficiency of distributed computing frame-work with the intelligence features of data exploration provided by NLP technologies. We introduce the technique of lattice querying which automatically forms the query from the set of text samples provided by a user by generalizing them in the level of parse trees. Also the system produces search results by matching parse trees of this query with that of candidate answers. Lattice queries allow increase in big data exploration efficiency since they form multiple "hypotheses" concerning user intent and explore data from multiple angles (generalizations).

Exploring data, mostly keyword query and phrase query are popular, as well as natural language-like ones. Users of search engines appreciate more and more 'fuzzy match' queries, which help to explore new areas where the knowledge of exact key-words is lacking. Using synonyms, taxonomies, ontologies and query expansions helps to substitute user keywords with the domain-specific ones to find what the system believes users are looking for (Galitsky 2003).

The idea of lattice query is illustrated in Fig. 1. Instead of a user formulating a query exploring a dataset, he or she proves a few samples (expressions of interest) so that the system formulates a query as an overlap (generalization) of these samples, applied in the form of a lattice (shown in bold on the bottom).

Proceeding from a keyword query to regexp or fuzzy one allows making search more general, flexible, assists in exploration of a new domain, as set of document with unknown vocabulary. What can be a further step in this direction? We introduce lattice queries, based on natural language expressions which are generalized into an actual query.

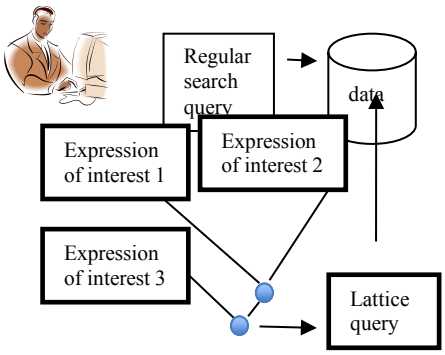

Fig. 1: Lattice query in comparison with regular queries

Nowadays, search engines ranging from open source to enterprise offer a broad range of queries with string character-based similarity. They include Boolean queries, span queries which restrict the distances between keywords in a document, regular expressions queries which allow a range of characters at certain positions, fuzzy match queries and more-like-this which allow substitution of certain characters based on string distances. Other kinds of queries allow expressing constraints in a particular dimension, such as geo-shape query. Proceeding from a keyword query to regexp or fuzzy one allows making search more general, flexible, assists in exploration of a new domain, as set of document with unknown vocabulary. What can be a further step in this direction? We introduce lattice queries, based on natural language expressions which are generalized into an actual query. Instead of getting search results similar to a given expression (done by 'more like this' query), we first build the commonality expression between all or subsets of the given sample expressions, and then use it as a query. A frame query includes words as well as attributes such as entity types and verb attributes.

Sentence-based Lattice Queries

Let us start with an employee search example. Let us imagine a company looking for the following individuals:
A junior sale engineer expert travels to customers on site.
A junior design expert goes to customer companies.
A junior software engineer rushes to customer sites.
Given the above set of samples, we need to form a job-search query which would give us candidates somewhat similar to what we are looking for. A trivial approach would be to just turn each sample into a query and attempt to find an exact match. However most of times it would not work, so such queries need to release some constraints. How to determine which constraints need to be dropped and which keywords are most important?

To do that, we apply generalization to the set of these samples. For the entities and attributes, we form the least general generalization. The seniority of the job (adjective) 'junior' will stay. The job activity (noun phrase) varies, so we generalize them into *<job-activity>*. The higher-level reference to the job is '*expert*' and is common for all three cases, so stays. The verb for job responsibility varies, so we use *<action>*, which can be further specified as *<moving_action>*, using verb-focused ontologies like VerbNet. To generalize the last noun phrase, we obtain the generalization *<customer, NP>*.
junior <any job activity> expert <action> customer-NP.
This is a lattice query, which is expected to be run against job descriptions and find the cases which are supposed to be most desired, according to the set of samples.
In terms of parse trees of the potential sentences to be matched with the lattice query, we rewrite it as
JJ-junior NP- NN-expert VP-* NN-customer NP-**
The lattice query read as *find me a junior something expert doing-something-with customer of-something*.
Now we show how this template can applied to accept/reject a candidate answer *Cisco junior sale representative expert flew to customers data centers*.

We represent the lattice query as a conjunction of noun phrases (NP) and verb phrases (VP) set:
[[NP [DT-a JJ-junior NN- NN-*], NP [NN*-customers]], [VP [VB-* TO-to NN*-customers]]]*
The first NP covers the beginning of the lattice query above, and the second NP covers the end. VP covers the second half of the lattice query starting from *doing-something...*
The generalization between the lattice query and a candidate answer is
[[NP [JJ-junior NN- NN-*], NP [NN*-customers]], [VP [VB-* TO-to NN*-customers]]]*
One can see that the NP part is partially satisfied (the article a does not occur in the candidate answer) and VP part is fully satisfied.

Here are the parse trees for three samples

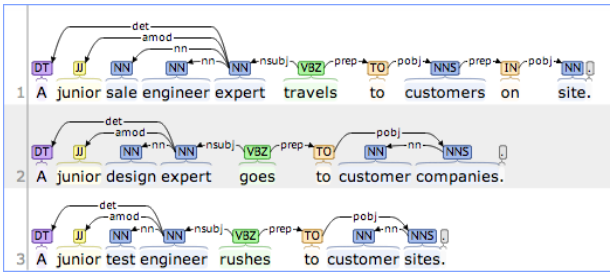

Fig. 2: Parse trees for three samples to form a lattice query

We obtain the lattice query to run against a dataset:
[[NP [DT-a JJ-junior NN- NN-*], NP [NN*-customers]], [VP [VB-* TO-to NN*-customers]]]*

One can see that using lattice queries, one can be very sensitive in selecting search results. Searching for a token followed by a word with certain POS instead of just a single token gives a control over false-positive rate. Automated derivation of such constraint allows user to focus on

cases instead of making efforts to generate a query which would keep expected search results in and unwanted out.
Definition: a lattice query Q is satisfied by a sentence S, if $Q \hat{} S = S$.
In practice a weak satisfaction is acceptable, where $Q \hat{} S \in S$, but there are constraints on the parts of the lattice query:

- A number of parts in $Q \hat{} S$ should be the same as in Q;
- All words (not POS-* placeholders) from Q should also be in $Q \hat{} S$.

Paragraph-level Lattice Queries

Text samples to form a lattice query can be typed, but also can be taken from text already written by someone. To expand the dimensionality of content exploration, samples can be paragraph-size texts (Galitsky 2014).

Let us consider an example of a safety-related exploration task, where a researcher attempts to find a potential reason for an accident. Let us have the following texts as incidents descriptions. These descriptions should be generalized into a lattice query to be run against a corpus of texts for the purpose of finding a root cause of a situation being described.

Crossing the snow slope was dangerous. They informed in the blog that an ice axe should be used. However, I am reporting that crossing the snow field in the late afternoon I had to use crampons.

I could not cross the snow creek since it was dangerous. This was because the previous hiker reported that ice axe should be used in late afternoon. To inform the fellow hikers, I had to use crampons going across the show field in the late afternoon.

As a result of generalization from two above cases, we will obtain a set of expressions for various ways of formulating commonalities between these cases. We will use the following snapshot of a corpus of text to illustrate how a lattice query is matched with a paragraph:

I had to use crampons to cross snow slopes without an ice axe in late afternoon this spring. However in summer I do not feel it was dangerous crossing the snow.

We link two phrases in different sentences since they are connected by a rhetoric relation based on *However …*

```
rel:     <sent=1-word=1..inform>     ===>
<sent=2-word=4..report>
From [<1>NP'They':PRP]
TO   [<4>NP'am':VBP,   NP'reporting':VBG,
<8>NP'the':DT,           <9>NP'snow':NN,
<10>NP'field':NN,         <11>NP'in':IN,
<12>NP'the':DT,         <13>NP'late':JJ,
<14>NP'afternoon':NN,     <15>NP'I':PRP,
<16>NP'had':VBD,          <17>NP'to':TO,
<18>NP'use':VB, <19>NP'crampons':NNS]
```

We are also linking phrases of different sentences based on communicative actions:

```
rel:     <sent=1-word=6..report>     ===>
<sent=2-word=1..inform>
From                    [<4>NP'the':DT,
<5>NP'previous':JJ, <6>NP'hiker':NN]
TO    [<1>NP'To':TO,    <2>NP'inform':VB,
<3>NP'the':DT,          <4>NP'fellow':JJ,
<5>NP'hikers':NNS]
```

As a result of generalizing two paragraphs, we obtain the lattice query:

```
[[NP [NN-ice NN-axe ], NP [DT-the NN-
snow NN-* ], NP [PRP-i ], NP [NNS-
crampons ], NP [DT-the TO-to VB-* ], NP
[VB-* DT-the NN-* NN-field IN-in DT-the
JJ-late NN-afternoon (TIME) ]], [VP
[VB-was JJ-dangerous ], VP [VB-* IN-*
DT-the NN-* VB-* ], VP [VB-* IN-* DT-
the IN-that NN-ice NN-axe MD-should VB-
be VB-used ], VP [VB-* NN-* VB-use ],
VP [DT-the IN-in ], VP [VB-reporting
IN-in JJ-late NN-afternoon (TIME) ], VP
[VB-* NN*-* NN-* NN*-* ], VP [VB-
crossing DT-the NN-snow NN-* IN-* ], VP
[DT-the NN-* NN-field IN-in DT-the JJ-
late NN-afternoon (TIME) ], VP [VB-had
TO-to VB-use NNS-crampons ]]]
```

Notice that potential safety-related "issues" are *ice-axe, snow, crampons, being at a … field during later afternoon, being dangerous, necessity to use ice-axe, crossing the snow*, and others. These issues occur in both samples, so that are of a potential interest. Now we can run the formed lattice query against the corpus and observe which issues extracted above are confirmed. A simple way to look at it is as a Boolean OR query: find me the conditions from the list which is satisfied by the corpus. The generalization for the lattice query and the paragraph above turns out to be satisfactory:

```
[[NP [NN-ice NN-axe ], NP [NN-snow NN*-* ],
NP [DT-the NN-snow ], NP [PRP-i ], NP [NNS-
crampons ], NP [NN-* NN-* IN-in JJ-late NN-
afternoon (TIME) ]], [VP [VB-was JJ-
dangerous ], VP [VB-* VB-use ], VP [VB-*
NN*-* IN-* ], VP [VB-crossing NN-snow NN*-*
IN-* ], VP [VB-crossing DT-the NN-snow ], VP
[VB-had TO-to VB-use NNS-crampons ], VP [TO-
to VB-* NN*-* ]]] => matched
```

Hence we got the confirmation from the corpus that the above hypotheses, encoded into this lattice query, are true. Notice that forming a data exploration queries from the original paragraphs would contain too many keywords and would produce too much marginally relevant results.

Generalization Operation

The purpose of an abstract generalization is to find the commonality between portions of text at various levels. The generalization operation occurs on the levels of *Article / Paragraph / Sentence /Phrase/Individual word.*

At each level except the lowest one, the result of the generalization of two expressions is a *set* of expressions. In such a set, expressions for which less-general expressions exist are eliminated. The generalization of two sets of expressions is a set of the sets that are the results of the pairwise generalization of these expressions.

Definition 1: Generalization of words

The result of generalization of two words,

$w_1 \wedge w_2 = < lemma(w_1) \wedge lemma(w_2), pos(w_1) \wedge pos(w_2) >$.

$lemma(w_1) \wedge lemma(w_2)$ is either $w_1 = w_2$, if the words are the same, or '*' otherwise (a placeholder for an arbitrary word, if words w_1 and w_2 are different.

$pos(w_1) \wedge pos(w_2)$ is either part-of-speech $pos(w_1) \wedge pos(w_2)$, or '*' otherwise (if their parts-of-speech are different).

Definition 2: Generalization of phrases:

- Only phrase of the same type can be generalized;
- For noun phrases, only those with the same head noun can be generalized. The generalization result includes this head noun. The head of a phrase is the word that determines the syntactic type of that phrase or analogously the stem that determines the semantic category of a compound of which it is a part. The rest of the words in phrases is generalized according to Definition 1.
- For verb phrases, they should have the same verb.
- For other types of phrases, generalization occurs analogously. Notice that English is primarily a head-initial language. Structure is descending as speech and processing move from left to right. Most dependencies have the head preceding its dependent(s), although there are also head-final dependencies in parse trees. For instance, the determiner-noun and adjective-noun dependencies are head-final as well as the subject-verb dependencies. Most other dependencies in English are, however, head-initial; the mixed nature of head-initial and head-final structures is common across languages.

Definition 3: Generalization of sentences.

$S_1 \wedge S_2 = \cup_p \cup_i p_{1i} \wedge p_{2j}$, where p_{1i} and p_{2j} are phrases from sentences S_1 and S_2 S_1 of type $p \in \{ NP, VP, ... \}$

We outline the algorithm for generalization two sentences below, which concerns paths of syntactic trees rather than sub-trees because these paths are tightly connected with language phrases. Regarding the operations on trees, we follow the work of (Kapoor & Ramesh 1995).

Although it is a formal operation on abstract trees, the generalization operation yields semantic information about the commonalities between sentences. Rather than extracting common keywords, the generalization operation produces a syntactic expression that can be semantically interpreted.

Evaluation

We conduct evaluation for complex information extraction tasks such as identifying communicative actions and detecting emotional states. Also, we perform evaluation for the rhetoric relation domain: this task is necessary to build a set of parse trees for a paragraph, linking its parse trees. We draw the comparison between information extraction based on the means available within Elasticsearch and Solr latticework:

- keyword Boolean queries,
- span queries where the distance between keywords in text is constrained, and
- lattice query-based information extraction.

The corpus is based on he set of customer complains, where both communicative actions and emotions are frequent and essential for complaint analysis tasks. Evaluation was conducted by quality assurance personnel.

Method / Task	Keywords and Regexps			Keywords and Regexp Queries			Span and 'Like' Queries			Lattice queries		
	P/R		speed	P/R		Speed	P/R		speed	P/R		speed
Communicative actions	64	71	1	63	72	0.02	68	70	0.05	82	75	15.1
Emotional state	62	70	1.2	59	70	0.02	64	68	0.05	80	74	18.2
Rhetoric relation	56	65	1.5	56	66	0.02	59	70	0.05	77	70	25.4

Table 1: Evaluation of lattice query-based information extraction tasks

Method Task	Forming lattice query as keyword overlap for two sentences	Forming lattice query as parse structure of a sentence	Lattice queries for two sentences	Forming lattice query as keyword overlap for paragraphs	Forming lattice query as parse structure	Lattice queries for two paragraphs
Legal research	59	62	70	43	51	62
Marketing research	55	68	69	46	53	64
Health research	52	65	71	42	55	67
Technology research	57	63	68	45	53	64
History research	60	65	72	42	52	65

Table 2: Evaluation of web mining via lattice queries

We observe in Table 1 that the information extraction F-measure for keywords and regular expressions is both 64% for querying indexed data and string search (although the former is about 50 times faster). Relying on span queries gives just 2% increase in F-measure, whereas using lattice queries delivers further 10% improvement.

We also evaluate the data exploration scenarios using search engine APIs. Instead of formulating a single complex question and submit it for search, a user is required to describe her situation in steps, so that the system would assist with formulating hypotheses on what is important and what is not. The system automatically derives generalizations and builds the respective set of lattice queries. Then search engine API is used to search the web with lattice queries and automatically filter out results which are not covered by the lattice query. To do the latter, the system generalizes each candidate search results with the lattice query and rejects the ones not covered, similar to the information extraction scenario.

This year I purchased my Anthem Blue Cross insurance through my employer. What is the maximum out-of-pocket expense for a family of two in case of emergency?

Last year I acquired my individual Kaiser health insurance for emergency cases only. How much would be my out of pocket expense within a year for emergency services for my wife and kids?

The system finds a commonality between these paragraphs and forms a lattice query, so that the search results are as close to this query as possible. An alternative approach is to derive a set of lattice queries, varying generalization results, and delivering those search results which are covered the best with one of the lattice query from this set (not evaluated here).

We show the percentage of relevant search results, depending on how queries are formed. We ran 20 queries for each evaluation setting and consider first 20 results for each. Each search results is consider as either relevant or not, and we do not differentiate between top search results and 15^{th}-20^{th} ones. We use Bing search engine API for these experiments. Evaluation of lattice querying on the web was conducted by the author.

One can see that for the sentence-level analysis, there is 14% improvement proceeding from keyword overlap to parse structures delivering phrases for web search, and further 8% improvement leveraging lattice queries derived from a pair of sentences. For the paragraphs, there are respective 21% and 22% improvement, since web search engine don't do well with paragraph-sized queries. If the number of keywords in a query is high, it is hard for a search engine to select which keywords are important, and term frequency becomes the major ranking factor. Also, for such queries, a search engine cannot rely on learned user selections from previous querying, hence the quality of search results are so low.

The proposed technique seems to be an adequate solution for cross-sentence alignment (Chambers et al 2007, MacCartney et al 2008). One application of this problem is automated solving of numerical equations formulated as algebra word problems (Kushman et al 2014). To form a representation for an elementary algebra problem *text*, we would use a training set of pairs *textT – equationT* and produce an alignment of *text* and *textT* by means of generalization *text ^ text* which is an expression which can be converted into a numerical expression. The capability to "solve" an algebraic problem is based on the completeness of a training set: for each type of equation, there should be a textual algebraic problem for it. Also, the problem of phrase alignment for such areas a machine translation has been explored in (Jiang and Conrath 1997).

In this work we introduced a new type of query for search engine framework, the lattice query, which is intended to facilitate the process of an abstract data exploration. Instead of having a user formulate a query, one or more instances are automatically formed from sample ex-

pressions. To derive a lattice query, as well as measure relevance of a question to an answer, an operation of syntactic generalization (Galitsky et al 2012) is used. It finds a maximal common sub-trees between the parse trees for the sample text fragments, and also it finds the maximum common sub-trees between the parse trees for the lattice query and that of the candidate answers. In the latter case, the size of the common sub-trees is a measure of relevance for a given candidate search results.

In our evaluation we compared the conventional information extraction approach where extraction rules are expressed using keywords and regular expressions, with the one where rules are frame queries. We observed that frame queries improve both precision and recall of information extraction by producing more sensitive rules, compared to sample expressions which would serve as extraction rules otherwise. For the web search, if one wants to find information relevant to a few portions of text, such as blog postings, Facebook reply or couple of articles of interest, lattice queries are a handy tool. It forms a web search (frame) query to find relevant results on the web and access their similarity. An importance of the lattice queries in data exploration is that only the most important keywords are submitted for web search, and neither single document nor keyword overlap deliver such the set of keywords.

Conclusions

In this work we introduced a new type of query for search engine framework, the lattice query, which is intended to facilitate the process of an abstract data exploration. Instead of having a user formulate a query, one or more instances are automatically formed from sample expressions. To derive a lattice query, as well as measure relevance of a question to an answer, an operation of syntactic generalization (Galitsky 2014) is used. It finds a maximal common sub-trees between the parse trees for the sample text fragments, and also it finds the maximum common sub-trees between the parse trees for the lattice query and that of the candidate answers. In the latter case, the size of the common sub-trees is a measure of relevance for a given candidate search results.

In our evaluation we compared the conventional information extraction approach where extraction rules are expressed using keywords and regular expressions, with the one where rules are frame queries. We observed that frame queries improve both precision and recall of information extraction by producing more sensitive rules, compared to sample expressions which would serve as extraction rules otherwise. For the web search, if one wants to find information relevant to a few portions of text, such as blog postings, Facebook reply or couple of articles of interest, lattice queries are a handy tool. It forms a web search (frame) query to find relevant results on the web and access their similarity. An importance of the lattice queries in data exploration is that only the most important keywords are submitted for web search, and neither single document nor keyword overlap deliver such the set of keywords.

The experimental environment, extended tree learning functionality, and the evaluation framework is available at http://code.google.com/p/relevance-based-on-parse-trees.

References

Borgida ER, McGuinness, DL. 1996. Asking Queries about Frames. Proceedings of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning 1996, 340—349.

MacCartney, B. Michel Galley, and Christopher D. Manning, 2008. A phrase-based alignment model for natural language inference. The Conference on Empirical Methods in Natural Language Processing (EMNLP-08), Honolulu, HI, October 2008.

Jiang JJ and D. W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. Proceedings of the International Conference on Research in Computational Linguistics.

DeNero J. and D. Klein. 2008. The Complexity of Phrase Alignment Problems. In Proceedings of ACL/HLT-08, pages 25–28.

Galitsky, B. 2003. Natural language question answering system: Technique of semantic headers. Advanced Knowledge International, Australia.

Galitsky, B., de la Rosa JL, Dobrocsi G. 2012. Inferring the semantic properties of sentences by mining syntactic parse trees. Data & Knowledge Engineering. Volume 81-82, November, 21-45.

Galitsky, B. 2012. *Machine Learning of Syntactic Parse Trees for Search and Classification of Text*. 2012. Engineering Application of AI, http://dx.doi.org/10.1016/j.engappai.2012.09.017.

Galitsky, B. 2014. Learning parse structure of paragraphs and its applications in search. Engineering Applications of Artificial Intelligence. 01/2014; 32:160–184.

Chambers, N, D. Cer, T. Grenager, D. Hall, C. Kiddon,. MacCartney, M. C. de Marneffe, D. Ramage, E. Yeh, and C. D. Manning. 2007. Learning Alignments and Leveraging Natural Logic. In Proceedings of the ACL-07 Workshop on Textual Entailment and Paraphrasing.

Kushman, N., Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to Automatically Solve Algebra Word Problems. ACL 2014.

Kapoor. S. and Ramesh, H. 1995.Algorithms for generating all spanning trees of undirected and weighted graphs, SIAM J. Comput., Vol. 24