Parse Thicket Representation for Multi-sentence Search

Boris A. Galitsky[1], Sergei O. Kuznetsov[2], and Daniel Usikov[3]

[1] eBay Inc San Jose CA USA
boris.galitsky@ebay.com
[2] Higher School of Economics, Moscow Russia
skuznetsov@hse.ru
[3] Dept. of Physics University of Maryland MD USA
usikov@hotmail.com

Abstract. We develop a graph representation and learning technique for parse structures for sentences and paragraphs of text. This technique is used to improve relevance answering complex questions where an answer is included in multiple sentences. We introduce Parse Thicket as a sum of syntactic parse trees augmented by a number of arcs for inter-sentence word-word relations such as coreference and taxonomic. These arcs are also derived from other sources, including Rhetoric Structure theory, and respective indexing rules are introduced, which identify inter-sentence relations and joins phrases connected by these relations in the search index. Generalization of syntactic parse trees (as a similarity measure between sentences) is defined as a set of maximum common sub-trees for two parse trees. Generalization of a pair of parse thickets to measure relevance of a question and an answer, distributed in multiple sentences, is defined as a set of maximal common sub-parse thickets. The proposed approach is evaluated in the product search domain of eBay.com, where user query includes product names, features and expressions for user needs, and the query keywords occur in different sentences of text. We demonstrate that search relevance is improved by single sentence-level generalization, and further increased by parse thicket generalization. The proposed approach is evaluated in the product search domain of eBay.com, where user query includes product names, features and expressions for user needs, and the query keywords occur in different sentences of text.

Keywords: learning taxonomy, learning syntactic parse tree, syntactic generalization, search relevance.

1 Introduction

The task of answering complex questions, where desired information is distributed through multiple sentences in a document, becomes the bottleneck of modern search engines. The demand for access to different types of information have led to a renewed interest in answering questions posed in ordinary human language and seeking exact, specific and complete answer. After having made substantial achievements in fact-finding and list questions, natural language processing (NLP) community turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organization, locations, dates, etc.) from single sentences in documents [4]. Complex questions often seek multiple different types of information simultaneously, located in multiple sentences, and do not presuppose that one single sentence could meet all of its information seeking expectations. To systematically analyze how keywords from query occur in multiple sentences in a document, one needs to explore coreferences and other relations between words within a sentence and between sentences.

Modern search engines attempt to find the occurrence of query keywords in a single sentence in a candidate search results [11]. If it is not possible or has a low search engine score, multiple sentences within one document are used. However, modern search engines have no means to determine if the found occurrences of the query keywords in multiple sentences are related to each other, to the same entity, and, being in different sentences, are all related to the query term.

In this study we attempt to systematically extract semantic features from paragraphs of text using a graph-based learning, assuming that an adequate parse trees for individual sentences are available. In our earlier studies [8,9] we applied graph learning to parse trees at the sentence level, and here we proceed to learning the structure of paragraphs, relying on *parse thickets*. Parse thicket is defined as a sum of parse trees with additional arcs between nodes for words in different sentences. We have defined the least general generalization of parse trees (we call it *syntactic generalization*), and in this study we extend it to the level of paragraphs. We propose parse thicket matching algorithm and apply it to re-rank multi-sentence answers to complex questions. Computing generalization of a pair of paragraph, we performed a pair-wise generalization for each sentence in paragraphs. This approach ignores the richness of coreference information, and in the current study we develop graph learning means specifically oriented to represent paragraphs of text as respective parse thickets with nodes interconnected by arcs for a number of relations including coreference. We consider a number of discourse-related theories such as Rhetoric Structure and Speech Acts as source of arcs to augment the parse thicket. These arcs will connect edges for words within as well as between parse trees for sentences.

Machine learning at the paragraph level is required for text classification problems, where handling the meaning (via collection of keywords) at the sentence level is insufficient, and taking advantage of coreference information is necessary [6]. In this paper we will demonstrate how building adequate paragraph structure is necessary when a paragraph is indexed for search. We will consider two cases for text indexing, where establishing proper coreferences inside and between sentences links entities in an index for proper match with a question (Fig. 1):

Text for indexing1: … Tuberculosis is usually a lung disease. It is cured by doctors specializing in pulmonology.

Text for indexing2: … Tuberculosis is a lung disease… Pulmonology specialist Jones was awarded a prize for curing a special form of disease.

Question: Which specialist doctor should treat my tuberculosis?

Fig. 1. Multi-sentence indexing cases

In the first case, establishing coreference link *Tuberculosis → disease → is cured by doctors pulmonologists* helps to match these entities with the ones from the question. In the second case this portion of text does not serve as a relevant answer to the question, although it includes keywords from this question. Hence at indexing time, keywords should be chained not just by their occurrence in individual sentences, but additionally on the basis of coreferences. If words X and Y are connected by a coreference relation, an index needs to include the chain of words X_0, X_1…X, Y_0,Y_1… Y, where chains X_0 , X_1…X and Y_0,Y_1… Y are already indexed (phrases including X and Y). Hence establishing coreference is important to extend index in a way to improve search recall. Notice that usually, keywords from different sentences can only be matched with a query keywords with a low score (high score is delivered by inter-sentence match).

Since our problem concerns with finding the best sentence that contains the answer to any given question, we need some mechanism that can measure how close the a candidate answer is to the question. This allows us to choose the final answer which is the one that matches the most closely to the question. To achieve this we need a representation of the sentences that allows us to capture useful information in order to accommodate the matching process. We also need an efficient matching process to work on the chosen representation.

The evaluation of matching mechanism in this study is associated with improvement of search relevance by checking syntactic similarity between query and sentences in search hits, obtained via a search engine API. This kind of syntactic similarity is important when a search query contains keywords which form a phrase, domain-specific expression, or an idiom, such as “shot to shot time”, “high number of shots in a short amount of time”. In terms of search implementation, this can be done in two steps:

1) Keywords are formed from query in a conventional manner, and search hits are obtained taking into account statistical parameters of occurrences these words in documents, popularity of hits, page rank and others.
2) Above hits are filtered with respect to syntactic similarity of the snapshots of search hits with search query. Parse thicket generalization comes into play here.

Hence we obtain the results of the conventional search and calculate the score of the generalization results for the query and each sentence and each search hit snapshot. Search results are then re-ranked and only the ones syntactically close to search query are assumed to be relevant and returned to a user.

2 Generalizing Portions of Text

To measure similarity of abstract entities expressed by logic formulas, a least-general generalization was proposed for a number of machine learning approaches, including explanation based learning and inductive logic programming. Least general generalization was originally introduced in [14]. Its realization within the predicate logic is opposite to the most general unification; therefore it is also called *anti-unification.* In this study, to measure similarity between portions of text such as paragraphs, sentences and phrases, we extend the notion of generalization from logic formulas to sets of syntactic parse trees of these portions of text. The purpose of an abstract generalization is to find commonality between portions of text at various semantic levels. Generalization operation occurs on the levels of Article/Paragraph/Sentence/Phrases (noun, verb and others)/Individual word.

At each level except the lowest one, individual words, the result of generalization of two expressions is a *set* of expressions. In such set, expressions for which there exist less general expressions are eliminated. Generalization of two sets of expressions is a set of sets which are the results of pair-wise generalization of these expressions.

We outline the algorithm for two sentences and then proceed to the specifics for particular levels (Fig. 2). The algorithm we present in this paper deals with paths of syntactic trees rather than sub-trees, because it is tightly connected with language phrases. We refer the reader to [8,9] for more details.

1) Obtain parsing tree for each sentence. For each word (tree node) we have the *word (lemma), part of speech* and *form of word* information. This information is contained in the node label. We also have an arc to the other node.
2) Split parse trees for sentences into sub-trees which are phrases for each type: *verb, noun, prepositional* and others; these sub-trees are overlapping. The sub-trees are coded so that information about occurrence in the full tree is retained.
3) All sub-trees are grouped by phrase types.
4) Extending the list of phrases by adding equivalence transformations
5) Generalize each pair of sub-trees for both sentences for each phrase type.
6) For each pair of sub-trees yield an alignment, and then generalize each node for this alignment. For the obtained set of trees (generalization results), calculate the score.
7) For each pair of sub-trees for phrases, select the set of generalizations with highest score (least general).
8) Form the sets of generalizations for each phrase types whose elements are sets of generalizations for this type.
9) Filtering the list of generalization results: for the list of generalization for each phrase type, exclude more general elements from lists of generalization for given pair of phrases.

Fig. 2. Sentence-level syntactic generalization algorithm

For a pair of phrases, generalization includes all *maximum* ordered sets of generalization nodes for words in phrases so that the order of words is retained. In the following example

To buy digital camera today, on Monday
Digital camera was a good buy today, first Monday of the month

Generalization is {<*JJ-digital, NN-camera*> ,<*NN- today, ADV-*, NN-Monday*>} where the generalization for noun phrases is followed by the generalization by adverbial phrase. Verb *buy* is excluded from both generalizations because it occurs in a different order in the above phrases. *Buy - digital - camera* is not a generalization phrase because *buy* occurs in different sequence with the other generalization nodes. Further details on sentence level generalization are available in [8].

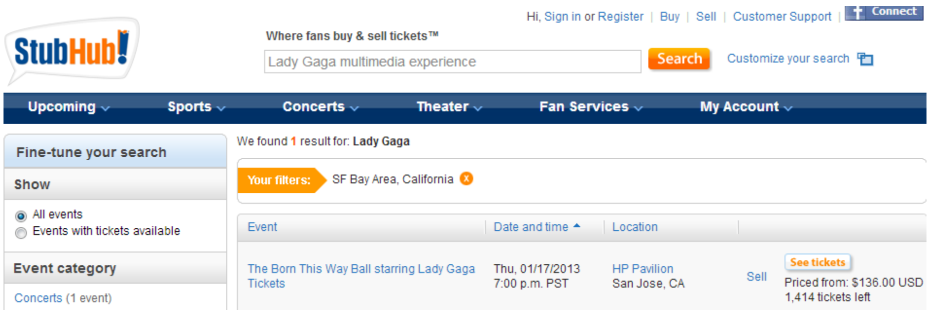
2.1 Direct Paragraph-Paragraph Match

We build a model of generalizing paragraphs taking into account coreference and taxonomic relationship between words between sentences, as well as within sentences. We will provide a number of examples to introduce the representation via parse thicket. We start with a simple example of how a discourse can be visualized by a forest.

> Lady Gaga has revealed that her next album will be released as an app.
> The singer confirmed that the album, called ARTPOP, will be a multimedia experience.
> She says she wants fans to "fully immerse" themselves in the project.
> Content will include extra music, videos, chat options and games.

To answer cross-sentence questions, we need to establish connections between the words of different sentences, taking into account that each consecutive sentence elaborate on the previous one.

Question "multimedia experience from lady gaga" (Fig. 3) will need the path in Parse Thiket *Lad_Gaga <possession>*$\longrightarrow$ *album <same entity relation>*$\longrightarrow$ *album <is-a relation>* $\rightarrow$ *multimedia_experience*. Question "Does Lady Gaga rely on games content" will involve the path "*Lad_Gaga <possession>*$\longrightarrow$ *album <has-a relation>*$\longrightarrow$ *content <has-a relation>*$\longrightarrow$ *games.*



Lady Gaga multimedia experience

We found 1 result for: Lady Gaga

Your filters: SF Bay Area, California

Event	Date and time	Location		
The Born This Way Ball starring Lady Gaga Tickets	Thu, 01/17/2013 7:00 p.m. PST	HP Pavilion San Jose, CA	Sell	See tickets Priced from: $136.00 USD 1,414 tickets left

Fig. 3. Parse Thicket- supported search for a concert at StubHub/eBay

To establish the semantic relationships above, we need to use multiple sources. Notice that we cannot rely on ontologies, only on syntactic information, searching in a horizontal domain. Within a sentence, we use its parse tree. *Same-entity relation* is based on anaphora resolution, and has-a relation is based on syntactic structure within a sentence. Between sentences, we use the elaboration assumption that each consecutive sentence elaborates on some entities from previous sentences. It turns out that Rhetoric Structure theory provides a systematic framework to do that.

We now show a paragraph which includes four sentences with the relations between the words. These relations can be established once taxonomy of domain entities is available [3]. In this Section we are interested in the structure of the paragraph, encoded by these relations. Below we will be representing and visualizing arcs for these relations together with edges of constituency parse tree. We use the relations *Same entity/Sub-entity (a partial case)/Super-entity (more general)/Sibling entity/New predicate for an entity.*

We can visualize information flow in a paragraph by just showing the structure of entities, without original sentences. Then it becomes clear how each sentence brings in a new form of constraint for the entities from the previous set of sentences in a paragraph. This structure is fairly important for answering a question: one needs to determine which level of specificity is best to answer it. The structure of relations must be taken into account indexing this paragraph for search in addition to keywords for each sentence. The best match between the parse thicket for a question (usually, a trivial parse tree) and the set of parse thickets for an answers does not only indicates the best answer, but also the most appropriate sentences within this answer according to desired specificity as expressed in the query.

Notice that the answer relevance to a question is measured by the cardinality of maximal common sub-thicket. In a conventional search engine, the closer the answer to the question, the higher the number of keywords common between the question and the answer (weighted according to TF*IDF model and according to distances between these keywords in the answer. Parse Thicket approach makes similarity measure more linguistically aware to the structure of text by means of forming maximum common sub-thicket.

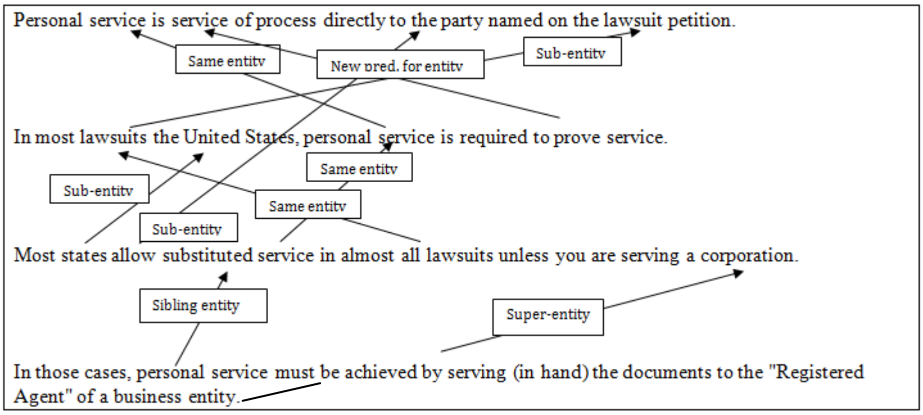

Personal service is service of process directly to the party named on the lawsuit petition.
Same entity
New pred. for entity
Sub-entity
In most lawsuits the United States, personal service is required to prove service.
Same entity
Sub-entity
Same entity
Sub-entity
Most states allow substituted service in almost all lawsuits unless you are serving a corporation.
Sibling entity
Super-entity
In those cases, personal service must be achieved by serving (in hand) the documents to the "Registered Agent" of a business entity.

Fig. 4. Parse thicket for a paragraph with *super/sub-entity/new predicate* relations

This parse thicket (Fig.4) is helpful to answer a question ‘How to serve a corporation in United States’ where we need to link the second and the third sentences via the nodes *service* and *lawsuit*.

In our third example, we visualize the discourse structure of customer review (Fig.5). We now draw the detailed constituency parse thicket and augment it by all possible arcs we discover for relations between words, including coreferences and sub-entities. The text to be represented as parse thicket is as follows:

> After numerous attempts to bring my parents into the digital world, I think I have finally succeeded. I failed a few years ago with a Sony digital camera that they could not quite figure out how to use and have succeeded only modestly with regards to the computer and internet surfing. But, heck I decided to give it another try when they asked about a digital camera the other day.

3 Extending Parse Thickets with Rhetoric Structure-Based Arcs

We have demonstrated how to build parse thicket based on coreference arcs and similar/related-words arcs. In this section we attempt to treat computationally, with a unified framework, two approaches to textual discourse:

- Rhetoric structure theory (RST [12]);
- Speech Act theory;

Although both these theories have psychological observation as foundations and are mostly of a non-computational nature, we will build a specific computational framework for them. We will use these theories to find links between sentences to enhance indexing for search. For the concept structure based formalization of Speech Act Theory, we refer to our earlier paper [7]. We proposed a graph-based mechanism to represent a structure of a dialog using nodes for communicative actions and edges for temporal and other relationships between them. We used a vocabulary of communicative actions to

1) find their subjects,
2) add respective arcs to the parse thicket,
3) index combination of phrases as subjects of communicative actions

For RST, we introduce explicit indexing rules which will be applied to each paragraph and:

1) attempt to extract an RST relation,
2) build corresponding fragment of the parse thicket, and
3) index respective combination of formed phrases (noun, verb, prepositional), including words from different sentences.

People sometimes assume that whenever a text has some particular kind of discourse structure, there will be a signal indicating that structure. A typical case would be a conjunction such as ‘but’.

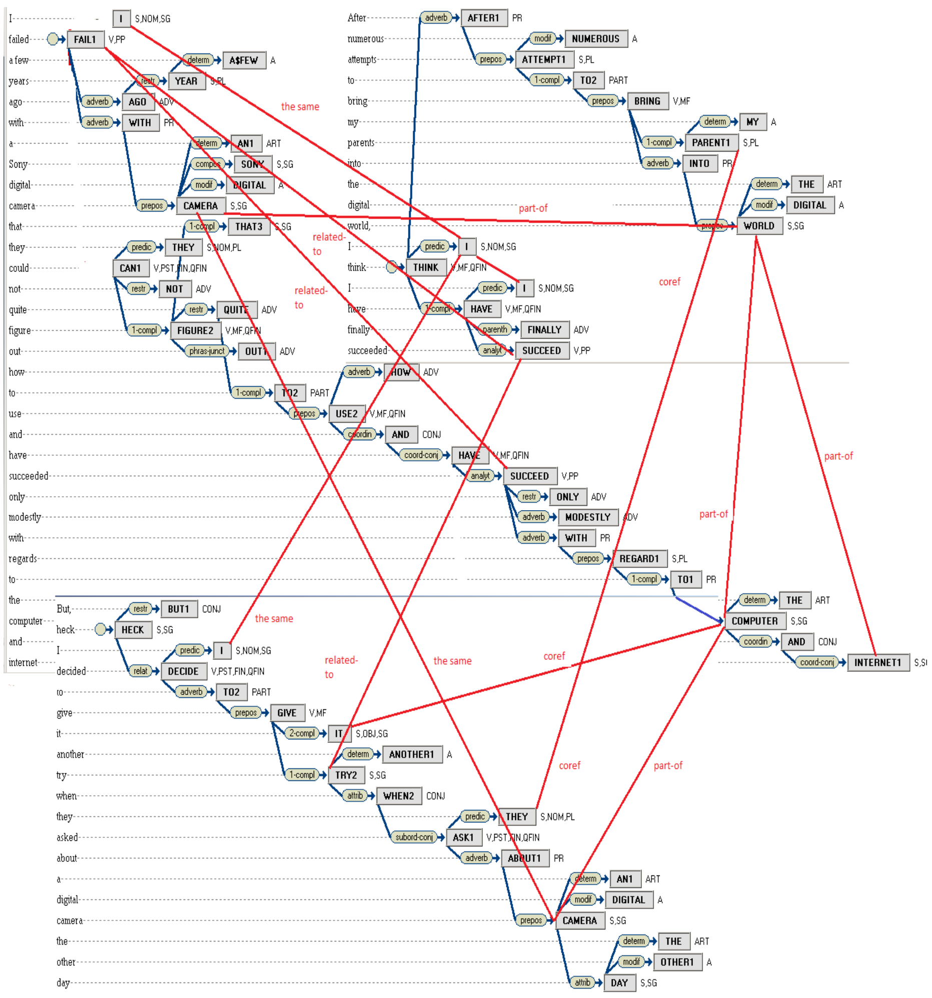


Fig. 5. Parse thicket for coreference, sub-entity and part-of relations

What structure is seen depends vitally on the words and sentences of the text are, but the relationship between words and text structure is extremely complex. Phrases and syntactic patterns can also be used to signal discourse structure. One expects discourse structure to be conveyed by signals. So the idea that discourse structure can be conveyed without signals is unexpected. Even more unexpected is the fact that for the discourse structure that RST represents, more than half is conveyed without explicit signals. On a relation-by-relation basis, it appears that every relation can be signaled in some contexts, and also that every relation can be conveyed without an explicit signal.

RST was originally developed as part of studies of computer-based text generation at Information Sciences Institute (part of University of Southern California) in about 1983 by Bill Mann, Sandy Thompson and Christian Matthiessen. The theory is designed to explain the coherence of texts, seen as a kind of function, linking parts of a text to each other. This coherence is explained by assigning a structure to the text, which slightly resembles a conventional sentence structure. We adjust this structure for the purpose of multi-sentence search ability.

We write *Syntactic template*

------------------------

Index(Part-of- Syntactic template)

where *Syntactic template* indicates how to extract a particular RST relation from text, in syntactic generalization format, and

Index(Part-of- Syntactic template) is a set of expressions which will be indexed in addition to the original sequence of words. *Part-of- Syntactic template* is a set of sub-lists of *Syntactic template.*

For the purpose of search, we build syntactic templates to express RST Relational classes. We don't have to cover all RST relation, and we don't have to be precise in establishing them, unless relation type is matched with query term. We give examples of some relations and respective templates we use to detect an RST links, and specify respective indexing rules for how to add additional joined phrases to the search index.

- Consequence (N/S), Result, Cause, Cause-Result

Nonvolitional-cause: ImperMentalVB ...NP. Maybe... VP

--

index(NP, VP), $\{$*remember, recall, notice*$\} \in$ *ImperMentalVB.*

In-response to NP ResponseVP

--

index(NP, ResponseVP)

NP AllowVP to ResultVP

--

index(NP, ResultVP), $\{$*allow, help, assist, give-ability*$\}\in$ *ResultVP*

- Manner, Means, Medium ('Medium demonstrates … feature … of the system')

MeansNP DemoVB NP to ToNP

--------------------------------------

index(NP, ToNP), $\{$*show, demonstrate, indicate*$\} \in$ *DemoVB*

- Temporal-before, Temporal-same-time, Temporal-after

VP until UntilVP

--------------------------------------

index(NP, UntilNP)

For the following text, we build parse thicket for RST (Fig. 6) and SpActT (Fig. 7).

Recently I tried to log into my account.
I received an error message that my account had been locked.
The site informed me to contact their appeal email address.
I have done so several times; however:
I get an email message back from Paypal stating that I cannot receive an answer of how to get into my account until I go to the site and login.
Well, this is impossible because my account is locked.

For the latter ParseThicket, we have the following structure of communicative actions which form the inter-sentence arcs in Fig.7:

try [log-into-my-account]→ receive [error-message]
 /
Inform [to-contact…]-
contact [their-appeal-email-address] → do [contact … however]
 /
*get [*email message back]
*state [*I cannot]→ *receive []*
 *answer [*how to …]
 get [get into my account until I go to the site and login]

We can now define a generalization operation on two parse thickets.

Given two parse thickets $C_x=(V_x, E_x)$ and $C_y=(V_y, E_y)$, generalization denoted C_x^C_y is defined as the set $\{G_1, G_2,…,G_k\}$ of all inclusion-maximal common subgraphs of C_x and C_y, such that each graph $G_i \in C_x$^$C_y = (V_x, E_x)$ is characterized as follows:

1) v_i is a vertex in G_i iff v_i is a vertex in both C_x and C_y which corresponds to CAs of the same party (opponent or proponent;
2) (v_x,v_y) is a thick (resp. thin) arc in G_i iff (v_i, v_j) is a thick (resp. thin) arc in C_x and C_y;
3) (v_x,v_y) is a thick (resp. thin) arc in G_i iff (v_i, v_j) is a thick (resp. thin) arc in C_x and (v_i, v_j) is a thin (resp. thick) arc in C_y
4) G_i contains at least one thick arc (v_i, v_j).

Note that when (v_i, v_j) is of the same type (thin or thick) in both C_x and C_y, then that type is adopted for (v_i, v_j) in G_i.

Condition 3) specifies that a thin arc (v_i, v_j). is adopted as an arc in G_i whenever there are arcs (v_i, v_j) in C_x and C_y of different types (thin arcs are seen thus as a weaker generalization of both thick and thin arcs).

By applying this definition of generalization we are now able to provide a criterion for accepting/rejecting an answer by generalizing it with the question and earlier approved/rejected answers. We outline a nearest neighbor approach to relating a new answer to the class of relevant/irrelevant answer classes, on the basis of its similarity with previous question-answer pairs in the training dataset.

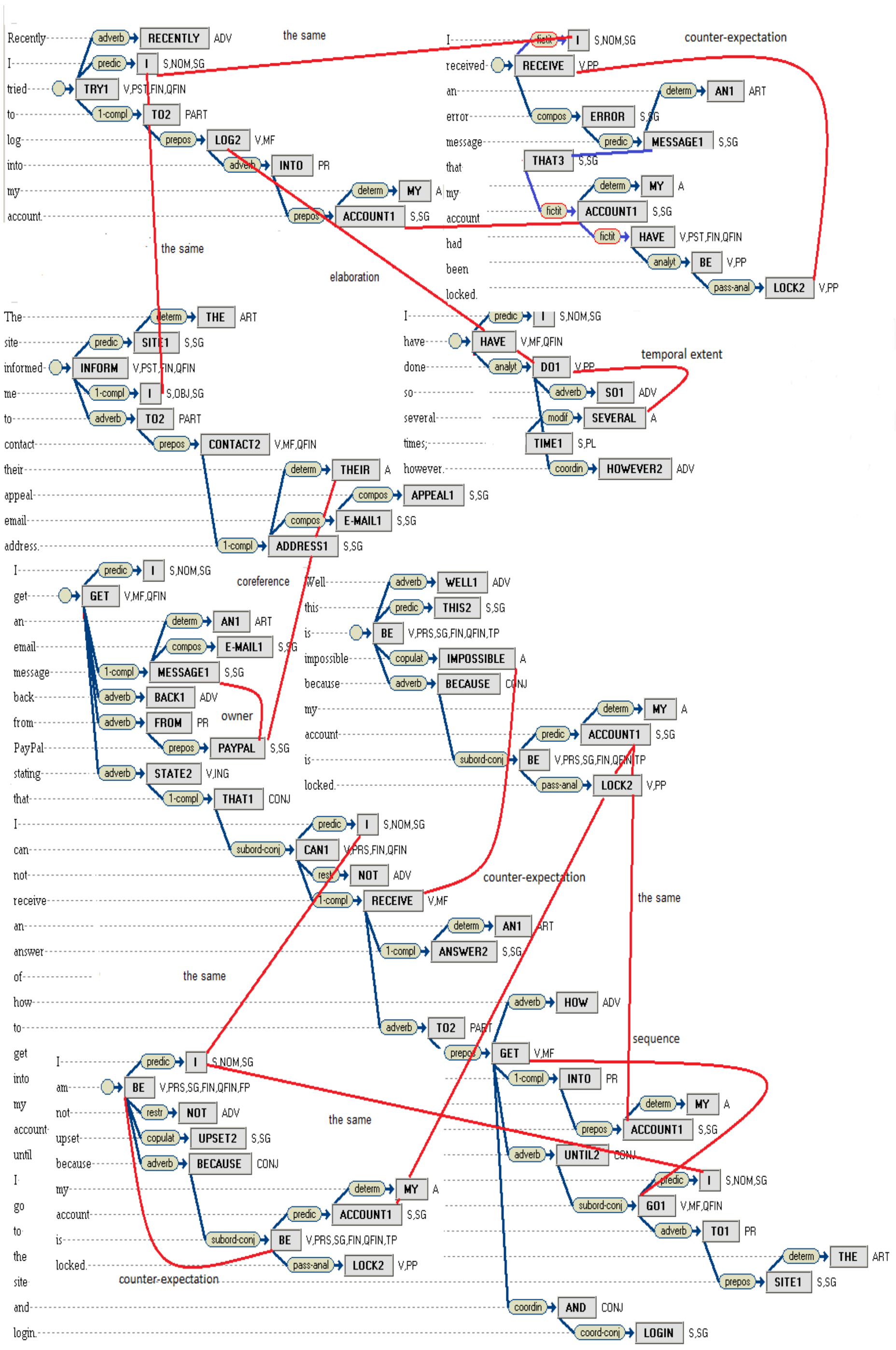

Recently I tried to log into my account
I received an error message that my account had been locked.
The site informed me to contact their appeal email address.
I have done so several times; however.
I get an email message back from PayPal stating that I can not receive an answer of how to get into my account until I go to the site and login.
Well this is impossible because my account is locked.
I am not upset because my account is locked.
the same
counter-expectation
the same
elaboration
temporal extent
coreference
owner
counter-expectation
the same
the same
sequence
the same
counter-expectation

Fig. 6. RST-based parse thicket

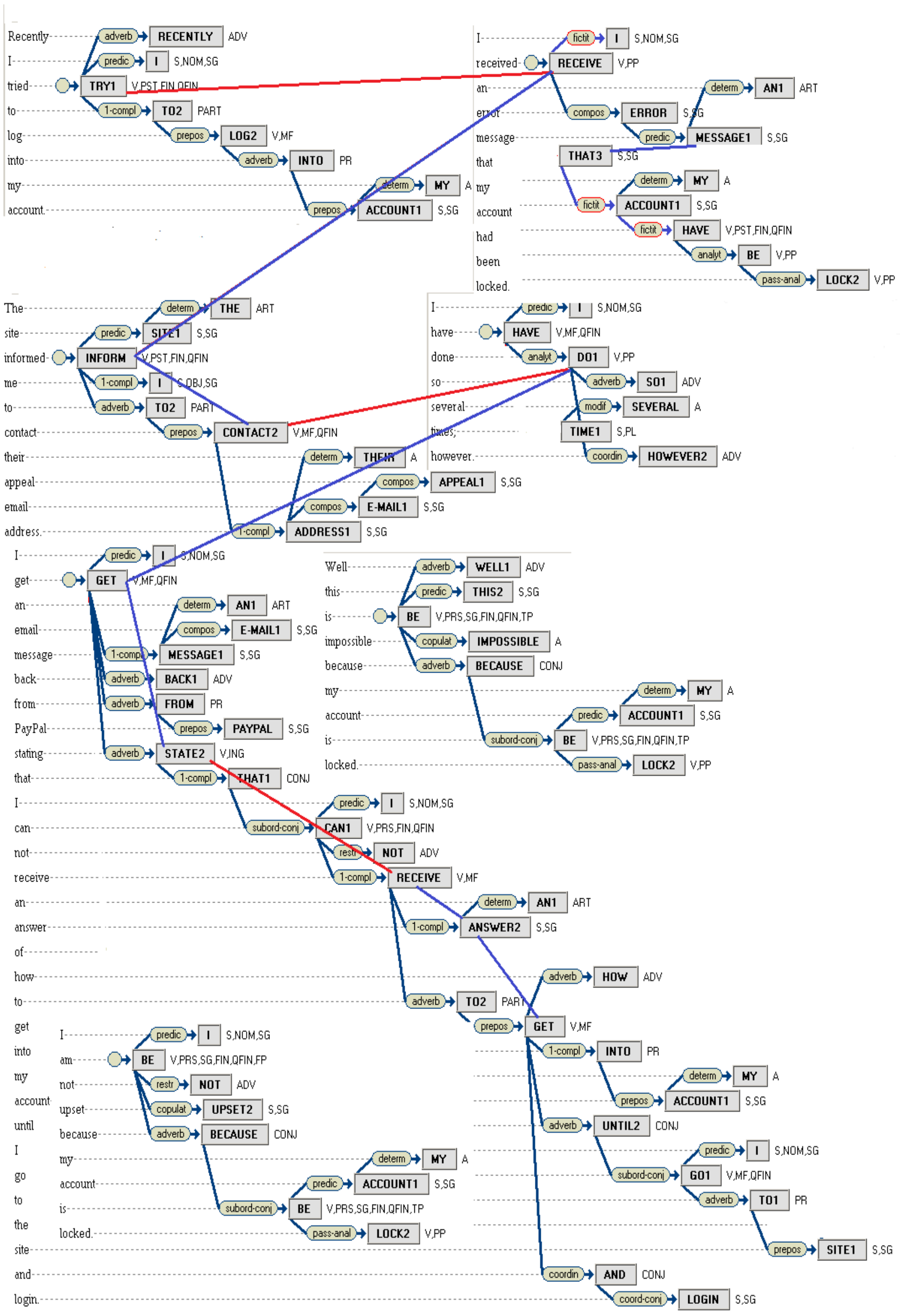

Recently I tried to log into my account.
I received an error message that my account had been locked.
The site informed me to contact their appeal email address.
I have done so several times; however.
I get an email message back from PayPal stating that I can not receive an answer of how to get into my account until I go to the site and login.
Well this is impossible because my account is locked.
I am not upset because my account is locked.

Fig. 7. SpActT-based parse thicket

Full parse thicket for the same paragraph we used for RST is depicted in Fig.7.

4 Evaluation of Parse Thicket Generalization

Syntactic generalization and parse thicket-based search has been implemented for entertainment-related domain at eBay's site StubHub.com. The query includes the desired performer, reference to a particular performance, as well as associated sentiments and feelings. Naturally, all such search criteria occur in different sentences, so the indexing system needs to find inter-sentence relations to verify that performers, events and user feelings are all properly related to each other.

The notion of query is rather broad in our case, including a posting in a blog, Facebook wall posting, or an email expressing an event attendance intent. The system is designed to answer complex queries about all products and associated sentiments, not just entertainment events. Queries are expected to include multiple sentences, where it is essential to track similarity between a query and abstract to improve user experience in search. In particular, the search is oriented to opinions data in linked aggregated form from various sources. To search for an opinion, a user specifies a product class, a name of particular products, a set of its features, specific concerns, needs or interests. A search can be narrowed down to a particular source, otherwise multiple sources of opinions (review portals, vendor-owned reviews, forums and blogs available for indexing) are combined. Search phrase may combine multiple sentences, for example: "*I am a beginner user of digital camera. I want to take pictures of my kids and pets. Sometimes I take it outdoors, so it should be waterproof to resist rain*". Obviously, this kind of specific opinion request can hardly be represented by keywords like 'beginner digital camera kids pets waterproof rain'.

We conducted evaluation of relevance of syntactic generalization – enabled search engine, based on Yahoo and Bing search engine APIs. For an individual query, the relevance was estimated as a percentage of correct hits among the first ten, using the values: {correct, marginally correct, incorrect} (compare with (Resnik, and Lin 2010)). Accuracy of a single search session is calculated as the percentage of correct search results plus half of the percentage of marginally correct search results. Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 40 search sessions.

For our evaluation, we use customers' queries to eBay entertainment and product-related domains, from simple questions referring to a particular product, a particular user need, as well as a multi-sentence forum-style request to share a recommendation. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the authors.

To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (syntactic generalization score and taxonomy-based score). To evaluate the performance of a hybrid system, we used the weighted sum of these two scores (the weights were optimized in an earlier search sessions).

Table 1 shows the search relevance evaluation results for single-sentence answers. The third and fourth columns show baseline Yahoo and Bing searches. The fifth column shows relevance of re-ranked search, and the last column shows relevance improvement compared with the baseline, the averaged Yahoo and Bing relevance.

Table 1. Evaluation of single-sentence search

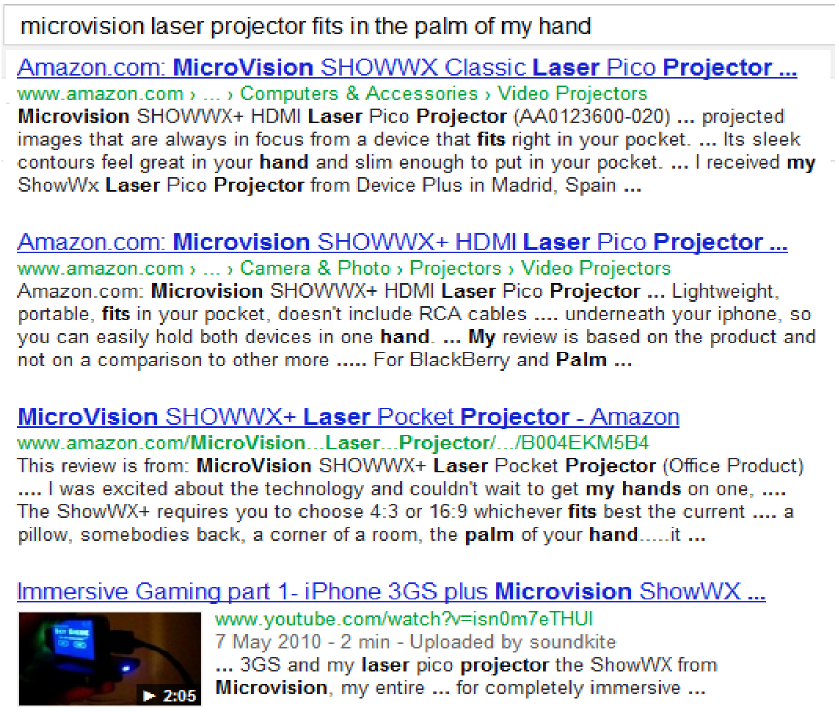
Query	phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by generalization, %, averaging over 40	Relevancy improvement: re-sorted relevance /(averaged for Bing & Yahoo)
3-4 word phrases	noun phrase	86.7	85.4	87.1	1.012
	verb phrase	83.4	82.9	79.9	0.961
	how-to expression	76.7	78.2	79.5	1.026
	Average	82.3	82.2	82.2	0.999
5-10 word phrases	noun phrase	84.1	84.9	87.3	1.033
	verb phrase	83.5	82.7	86.1	1.036
	how-to expression	82.0	82.9	82.1	0.996
	Average	83.2	83.5	85.2	1.022
2-3 sentences	one verb one noun phrases	68.8	67.6	69.1	1.013
	both verb phrases	66.3	67.1	71.2	1.067
	one sent of how-to type	66.1	68.3	73.2	1.089
	Average	67.1	67.7	71.2	1.056

We observe that using syntactic generalization improves the relevance of search in cases where query is relatively complex. For shorter sentences there is a slight drop in accuracy (-0.1%), for medium-length queries of 5-10 keywords we get 2% improvement, and 5.6% improvement for multi-sentence query. As the absolute performance of search naturally drops when queries become more complex, relative contribution of syntactic generalization increases.

We did not find a significant correlation between a query type, phrase type, and search performance with and without syntactic generalization for these types of phrases. Verb phrases in questions did well for multi-sentence queries perhaps because the role of verbs for such queries is more significant than for simpler queries where verbs can be frequently ignored.

Modern search engines attempt to find the occurrence of query keywords in a single sentence in a candidate search results. If it is not possible or has a low search engine score, multiple sentences within one document are used. However, modern search engines have no means to determine if the found occurrences of query keywords:

Are related to each other / Are related to the same entity /Being in different sentences, all related to the query term.



microvision laser projector fits in the palm of my hand

Amazon.com: **MicroVision** SHOWWX Classic **Laser** Pico **Projector ...**
www.amazon.com › ... › Computers & Accessories › Video Projectors
Microvision SHOWWX+ HDMI **Laser** Pico **Projector** (AA0123600-020) **...** projected images that are always in focus from a device that **fits** right in your pocket. **...** Its sleek contours feel great in your **hand** and slim enough to put in your pocket. **...** I received **my** ShowWx **Laser** Pico **Projector** from Device Plus in Madrid, Spain **...**

Amazon.com: **Microvision** SHOWWX+ HDMI **Laser** Pico **Projector ...**
www.amazon.com › ... › Camera & Photo › Projectors › Video Projectors
Amazon.com: **Microvision** SHOWWX+ HDMI **Laser** Pico **Projector ...** Lightweight, portable, **fits** in your pocket, doesn't include RCA cables .... underneath your iphone, so you can easily hold both devices in one **hand**. **... My** review is based on the product and not on a comparison to other more ..... For BlackBerry and **Palm ...**

MicroVision SHOWWX+ **Laser** Pocket **Projector** - Amazon
www.amazon.com/**MicroVision**...**Laser**...**Projector**/.../B004EKM5B4
This review is from: **MicroVision** SHOWWX+ **Laser** Pocket **Projector** (Office Product) **....** I was excited about the technology and couldn't wait to get **my hands** on one, **....** The ShowWX+ requires you to choose 4:3 or 16:9 whichever **fits** best the current **....** a pillow, somebodies back, a corner of a room, the **palm** of your **hand**.....it **...**

Immersive Gaming part 1- iPhone 3GS plus **Microvision** ShowWX ...
www.youtube.com/watch?v=isn0m7eTHUI
7 May 2010 - 2 min - Uploaded by soundkite
... 3GS and my **laser** pico **projector** the ShowWX from **Microvision**, my entire **...** for completely immersive **...**

Fig. 8. Search results for the query requiring PARSE THICKET to provide relevance answers

To illustrate this statement, we search Google for 'microvision laser projector which fits in the palm of my hand'.

The expected/desired answer is as follows:

http://www.popularmechanics.com/technology/gadgets/4244056   I also saw another projection technology yesterday that looked pretty close to production. A company called **Microvision** produces a tiny, portable **projector** that uses red, green and blue **lasers** and a single tiny micromirror to project an image much the way old cathode ray tube televisions did, by scanning lines to create 60 frames per second. The whole **projector fits in the palm of your hand**, but a lot of that size comes from the battery; if you had an external power source, such as a USB, this thing could be as small as your thumb.   Read more: Innovative Projectors Will Fit in Your Palm, Cellphone: Buzzword @ CES 2008 (With Video) - Popular Mechanics

First few hits obtained by Google are shown in Fig 8. We observe all of the above tree problems in each of the search result. All answers are indeed about a 'Microvison projector', but user needs is represented by neither search result snippet. And if the keywords from the user need part of the query occur, they are not related to the main entity, 'Microvison projector'.

One can see that 'Microvision' (company name), 'laser' (type of product), 'fits in the palm of my hand' (user need) are most likely occur in different sentences, so matching of parse thicket is required to find a document with relevant information. Parse thicket approach would work best at indexing time, however in this paper we evaluate search relevance improvement in horizontal domain, re-ranking search engine API results.

4.1 Evaluation of Multi-sentence Search

To conduct a multi-sentence search evaluation, we also use Yahoo and Bing search APIs as for the single-sentence answers. We selected queries from eBay product searches which included reference to a product and a number of user need. Frequently expressions for these needs occurred in multiple sentences in product reviews, shopping forums and blogs. We also automatically filtered out the cases which gave satisfactory one-sentence answers to build multi-sentence parse thicket-based evaluation set.

Discovering trivial (in terms of search relevance) links between different sequences (such as coreferences) is not as important for search as finding more implicit links provided by text discourse theories. We separately measure search relevance when parse thicket is RST-based and SpActT-based. Since these theories are the main sources for establishing non-trivial links between sentences, we limit ourselves to measuring the contributions of these sources of links. Our hybrid approach includes both these sources for links.

We now conduct specific evaluation where answers are distributed through two or more sentences. If it is not the case, we exclude a query from our evaluation set. We consider all cases of questions (phrase, one, two, and three sentences) and all cases of search results occurrences (compound sentence, two, and three sentences) and measured how parse thicket improved the search relevance, compared to original search results ranking averaged for yahoo and Bing.

One can see that even the simplest cases of short query and a single compound sentence gives more than 5% improvement. Parse thicket - based relevance improvement stays within 7-9% as query complexity increases by a few keywords, and then increases to 9-11% as query becomes one-two sentences. For the same query complexity, naturally, search accuracy decreases when more sentences are required for answering this query. However, contribution of the parse thicket does not vary significantly with the number of sentences the answer occurs in.

Table 2. Search improvement results for parse thicket approach

Query	Answer	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by pair-wise sentence generalization, %, averaging over 40 searches	Relevancy of re-sorting by forest generalization based on RST, %, averaging over 20 searches	Relevancy of re-sorting by forest generalization based on SpActT, %, averaging over 20 searches	Relevancy of re-sorting by hybrid RST+SpActT forest generalization, %, averaging over 40 searches	Relevancy improvement for parse thicket approach, comp. to pair-wise generalization
3-4 word phrases	1 comp. sentence	81.7	82.4	86.6	88.0	87.2	91.3	1.054
	2 sent	79.2	79.9	82.6	86.2	84.9	89.7	1.086
	3 sent	76.7	75.0	79.1	85.4	86.2	88.9	1.124
	Average	79.2	79.1	82.8	86.5	86.1	90.0	1.087
5-10 word phrases	1 comp. sentence	78.2	77.7	83.2	87.2	84.5	88.3	1.061
	2 sent	76.3	75.8	80.3	82.4	83.2	87.9	1.095
	3 sent	74.2	74.9	77.4	81.3	80.9	82.5	1.066
	Average	76.2	76.1	80.3	83.6	82.9	86.2	1.074
1 sentence	1 comp. sent	77.3	76.9	81.1	85.9	86.2	88.9	1.096
	2 sent	74.5	73.8	78.	82.5	83.1	86.3	1.101
	3 sent	71.3	72.2	76.5	80.7	81.2	83.2	1.088
	Average	74.4	74.3	78.7	83.0	83.5	86.1	1.095
2 sentences	1 comp. sent	75.7	76.2	82.2	87.0	83.2	83.4	1.015
	2 sent	73.1	71.0	76.8	82.4	81.9	82.1	1.069
	3 sent	69.8	72.3	75.2	80.1	79.6	83.3	1.108
	Average	72.9	73.2	78.1	83.2	81.6	82.9	1.062
3 sentences	1 sentence	73.6	74.2	78.7	85.4	83.1	85.9	1.091
	2 sentences	73.8	71.7	76.3	84.3	83.2	84.2	1.104
	3 sentences	67.4	69.1	74.9	79.8	81.0	84.3	1.126
	Average	71.6	71.7	76.6	83.2	82.4	84.8	1.107
Average for all Query and Answer type								1.085

Notice that there is a noticeable improvement of accuracy in the comparable cases between Tables 1 and 2. While single-sentence syntactic match gives 5.6% improvement, multi-sentences parse thickets provides 8.7% for the comparable query complexity (5.4% for single-sentence answer) and up to 10% for the cases with more complex answers. One can see that parse thicket improves single sentence syntactic generalization by at least 2%. On average through the cases of Table 2, parse thickets outperforms single sentence syntactic generalization by 6.7%, whereas RST on its own gives 4.6% and SpActT-4.0% improvement respectively. Hybrid RST + SpActT gives 2.1% improvement over RST-only and 2.7% over SpActT only. We conclude that RST links compliment SpActT links to properly establish relations between entities in sentences for the purpose of search.

5 Related Work and Conclusions

Usually, classical approaches to semantic inference rely on complex logical representations. However, practical applications usually adopt shallower lexical or lexical-syntactic representations, but lack a principled inference framework. [2] proposed a generic semantic inference framework that operates directly on syntactic trees. New trees are inferred by applying entailment rules, which provide a unified representation for varying types of inferences. The current work deals with syntactic tree transformation in the graph learning framework, treating various phrasings for the same meaning in a more unified and automated manner.

The set of semantic problems addressed in this paper is of a much higher semantic level compared to SRL, therefore more sensitive tree matching algorithm is required for such semantic level. In terms of this study, semantic level of classification classes is much higher than the level of semantic role labeling or semantic entailment. SLR does not aim to produce complete formal meanings, in contrast to our approach. Unlike [19] who uses edit distance for finding optimal dependency tree matching, we use maximal set of common sub-graphs which obeys logical properties of least general generalization and is therefore better suited to ascend to semantic level (of logical forms representation). This study operates on the level of paragraphs instead of sentences and our previous studies [7, 8]. Also, we apply re-ranking to search engine results and not a raw index. Lexical chain formalism can be considered as a special case of parse thicket. Paper [5] considered keywords as condensed versions of documents and short forms of their summaries. The authors treat the problem of automatic extraction of keywords from documents as a supervised learning task.

In this study we introduced the notion of syntactic generalization to learn from parse trees for a pair of sentences, and extended it to learning augmented parse thickets for two paragraphs. Parse thicket is intended to represent syntactic structure of text as well as a number of semantic relations for the purpose of indexing for search. To accomplish this, parse thicket includes relations between words in different sentences, such that these relations are essential to match queries with portions of texts to serve as an answer.

We considered the following sources of relations between words in sentences:

Coreferences, Taxonomic relations such as sub-entity, partial case, predicate for subject etc.; Rhetoric structure relation and Speech acts. Since the first and second source of relations has been explored in details, we focus our evaluation on the contribution of third and fourth sources. We demonstrated that search relevance can be improved, if search results are subject to confirmation by sentence-level syntactic generalization, if answer occurs in a single sentence, and by parse thicket generalization, if answer occurs in multiple sentences.

Traditionally, machine learning of linguistic structures is limited to keyword forms and frequencies. At the same time, most theories of discourse are not computational, they model a particular set of relations between consecutive states. In this work we attempted to achieve the best of both worlds: learn complete parse tree information augmented with an adjustment of discourse theory allowing computational treatment.

Graphs have been used extensively to formalize ranking of NL texts [18]. Graph-based ranking algorithms are a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. Using semantic information for query ranking has been proposed in [1]. Moreover, relying on matching of parse trees of a question and an answer has been the subject of [13]. However, we believe the current study leads the way in multi-sentence relevance improvement, relying on learning parse trees and linguistic theories of the nature of the coherence of texts.

Acknowledgments. The second author was supported by the project "Mathematical Models, Algorithms, and Software Tools for Intelligent Analysis of Structural and Textual Data" within the framework of the Basic Research Program at the National Research University Higher School of Economics, Moscow, Russia.

References

1. Aleman-Meza, B., Halaschek, C., Arpinar, I., Sheth, A.: A Context-Aware Semantic Association Ranking. In: Proc. First Int'l Workshop Semantic Web and Databases (SWDB 2003), pp. 33–50 (2003)
2. Bar-Haim, R., Dagan, I., Greental, I., Shnarch, E.: Semantic Inference at the Lexical-Syntactic Level AAAI (2005)
3. Bhogal, J., Macfarlane, A., Smith, P.: A review of ontology based query expansion. Information Processing & Management 43(4), 866–886 (2007)
4. Chali, Y., Hasan, S.A., Joty, S.R.: Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. Inf. Process. Manage. 47(6), 843–855 (2011)
5. Ercan, G., Cicekli, I.: Using lexical chains for keyword extraction. Information Processing & Management 43(6), 1705–1714 (2007)
6. Galitsky, B.: Natural Language Question Answering System: Technique of Semantic Headers. Advanced Knowledge International, Australia (2003)
7. Galitsky, B., González, M.P., Chesñevar, C.I.: A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. Decision Support Systems 46(3), 717–729 (2009)

8. Galitsky, B., Dobrocsi, G., de la Rosa, J.L.: Inferring semantic properties of sentences mining syntactic parse trees. Data & Knowledge Engineering 81-82, 21–45 (2012)
9. Galitsky, B., Dobrocsi, G., de la Rosa, J.L., Kuznetsov, S.O.: Using Generalization of Syntactic Parse Trees for Taxonomy Capture on the Web. In: 19th International Conference on Conceptual Structures, ICCS 2011, pp. 104–117 (2011)
10. Kapoor, S., Ramesh, H.: Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs. SIAM J. Computing 24, 247–265 (1995)
11. Kim, J.-J., Pezik, P., Rebholz-Schuhmann, D.: MedEvi: Retrieving textual evidence of relations between biomedical concepts from Medline. Bioinformatics 24(11), 1410–1412 (2008)
12. Mann, W.C., Christian, M.I., Matthiessen, M., Thompson, S.A.: Rhetorical Structure Theory and Text Analysis. In: Mann, W.C., Thompson, S.A. (eds.), pp. 39–78. John Benjamins, Amsterdam (1992)
13. Moschitti, A.: Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 318–329. Springer, Heidelberg (2006)
14. Plotkin, G.D.: A note on inductive generalization. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 5, pp. 153–163. Elsevier North-Holland, New York (1970)
15. Punyakanok, V., Roth, D., Yih, W.: The Necessity of Syntactic Parsing for Semantic Role Labeling. In: IJCAI (2005)
16. OpenNLP (2012), `http://incubator.apache.org/opennlp/documentation/manual/opennlp.html`
17. Marcu, D.: From Discourse Structures to Text Summaries. In: Mani, I., Maybury, M. (eds.) Proceedings of ACL Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, pp. 82–88 (1997)
18. Mihalcea, R., Tarau, P.: TextRank: Bringing Order into Texts. In: Empirial Methods in NLP (2004)
19. Punyakanok, V., Roth, D., Yih, W.: Mapping dependencies trees: an application to question answering. In: Proceedings of AI & Math., Florida, USA (2004)