Learning Parse Structure of Paragraphs and its Applications in Search

Boris Galitsky

Knowledge-Trail Inc. San Jose CA 95127

Abstract

We propose to combine parse forest and discourse structures to form a unified representation for a paragraph of text. The purpose of this representation is to tackle answering complex paragraph-sized questions in a number of products and services-related domains. A candidate set of answers, obtained by a keyword search, is re-ranked by matching the sequence of parse trees of an answer with that of the question. To do that, a graph representation and learning technique for parse structures for paragraphs of text have been developed. Parse Thicket (PT) as a set of syntactic parse trees augmented by a number of arcs for inter-sentence word-word relations such as co-reference and taxonomic relations is introduced. These arcs are also derived from other sources, including Speech Act and Rhetoric Structure theories. The operation of generalization of logical formulas is extended towards parse trees and then towards parse thickets to compute similarity between texts.

We provide a detailed illustration of how PTs are built from parse trees, and generalized. The proposed approach is subject to evaluation in the product search and recommendation domain of eBay.com, where user queries include product names, desired features and expressions for user needs in multiple sentences. We demonstrate that search relevance is improved by PT generalization, using Bing search engine API as a baseline. We perform the comparative analysis of contribution of various sources of discourse information to the relevance. An open source plugin for SOLR is developed so that the proposed technology can be easily integrated with industrial search engines.

Keywords: Parse Forest, Parse Thicket, Graph Learning

1. Introduction

Parse trees were found fairly useful in solving text classification and text relevance problems, including search. Parse trees have become a standard form of representing the syntactic structures of sentences (Abney, 1991; Punyakanok *et al.*, 2005). A number of approaches to learning parse trees have been proposed, including convolution kernel based on support vector (SVM) learning (Collins and Duffy, 2002; Haussler, 1999; Moschitti, 2006), and structural similarity based on direct matching of parse trees for sentences (Galitsky et al 2012).

As to the structure of a paragraph of text, there is no generally accepted model. Such a model needs to rely on the parse forest for the sentences this paragraph comprises, and also need to include a paragraph-level discourse information. In this study we will attempt to represent a linguistic structure of a paragraph of text based on parse trees for each sentence of this paragraph. We will refer to the sequence of parse trees plus a number of arcs for inter-sentence relations of the discourse type between the nodes for words as *Parse Thicket* (PT). A PT is a graph, which includes parse trees for each sentence, as well as additional arcs for inter-sentence relationship. Parse thickets are inspired by the problem of answering complex paragraph-size questions, which needs to be matched with paragraphs of candidate answers.

We explore a number of sources for the links between words in a paragraph other than syntactic. Our sources are coreferences, entity-entity and other taxonomic relations, discourse relations such as rhetoric relations between elementary discourse units, and speech act relations. For each of these sources we determine whether it can be leveraged by more accurate assessment of similarity between paragraphs of

text. The main goal here is to make similarity measure independent of how text / phrases is distributed through sentences.

A vast majority of linguistic theories, from cognitive to functional theories of grammars, rely on need some form of representation of structured data for natural language processing (Lamberti et al 2009). Once a linguistic representation becomes richer than the bag-of-words, there is a need for a systematic way to compare such representations. However, there is a lack of formal models to compare linguistic structures beyond parse trees for individual sentences. In this study we introduce PTs as a structural machine learning framework to operate with multiple syntactic parse trees for sentences in *paragraphs* of text.

1.1 Answering paragraph-size questions

Nowadays, commercial search engines are not very good at tackling paragraph-level queries consisting of multiple sentences. They either find very similar documents, if they are available, or very dissimilar ones, so that search results are not very useful to the user. This is due to the fact that for multi-sentences queries it is rather hard to learn ranking based on user clicks, since the number of longer queries is practically unlimited. Hence we need a linguistic technology, which would rank candidate answers based on structural similarity between the question and the answer. In this study we build a graph-based representation for a paragraph of text so that we can track the structural difference between these paragraphs, taking into account not only parse trees, but the whole discourse of both the question and answers

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 (Chali et al 2009). Unlike simple factoid questions, complex questions often seek multiple different types of information simultaneously, located in multiple sentences, and one cannot assume that a particular single sentence contains expected information. Dependency parsing helps to answer complex queries in an industrial environment, reconstructing semantic content efficiently by extracting related terms through analysis and classification (Iwashita et al 2011). To systematically analyze how keywords from a query occur in multiple sentences in a document, one needs to explore coreferences and other relations between words within a sentence and between sentences.

Most search engines attempt to find the occurrence of query keywords in a single sentence in a candidate search result (Kim et al 2006). 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 or not. Neither search engine can determine if they are related to the same entity, nor, being in different sentences, are all related to the query term.

Paragraphs of text as queries appear in the search-based recommendation domains (Montaner et al., 2003; Bhasker and Srikumar 2010; Thorsten, 2012). Recommendation agents track user chats, user postings on blogs and forums, user comments on shopping sites, and suggest web documents and their snippets, relevant to a purchase decisions. To do that, these recommendation agents need to take portions of text, produce a search engine query, run it against a search engine API such as Bing or Yahoo, and filter out the search results which are determined to be irrelevant to a purchase decision. The last step is critical for a sensible functionality of a recommendation agent, and poor relevance would lead to a lost trust in the recommendation engine. Hence an accurate assessment of similarity between two portions of text is critical to a successful use of recommendation agents.

Nowadays, the problem of answering simple queries, involving a single entity and its attributes, is solved fairly well. However, more complex questions in such domain as legal, science, and health can be expressed in paragraphs rather than in single phrases or single sentences, as expressed, for example, in Yahoo! Answers or StackOverflow. The technique being proposed targets the paragraph-sized questions which involve multiple entities and their inter-connected attributes. These attributes do not occur altogether but instead follow a certain discourse, which needs to be extracted from the question and then matched with

the ones from candidate answers. For example, to answer the following question, we do not just need to match the keywords from actual questions (two last sentences) but also match the preceding sentences with that of a candidate answer.

> One of the provisions in the Patient Protection and Affordable Care Act is that it limits the profits of health insurance companies. The ACA imposes a minimum medical loss ratio (MLR) on all insurers. The MLR is the amount of money spent on covered person medical care divided by the total revenue received through premiums. What constitutes 'medical care'? Do investments in electronic health records count as medical care?

If a question includes just a phrase or a sentence, selecting less frequent keywords to match with candidate answers do the job. However, in this case one needs to track how entities (like 'medical care') are introduced and how the links between their attributes are established. To match this paragraph-sized question with an answer, one needs to build a representation of its discourse on one hand (PT), and also provide a machinery to match this discourse structure with that of a candidate answer (PT generalization).

1.2 From sentence-level to paragraph-level generalization

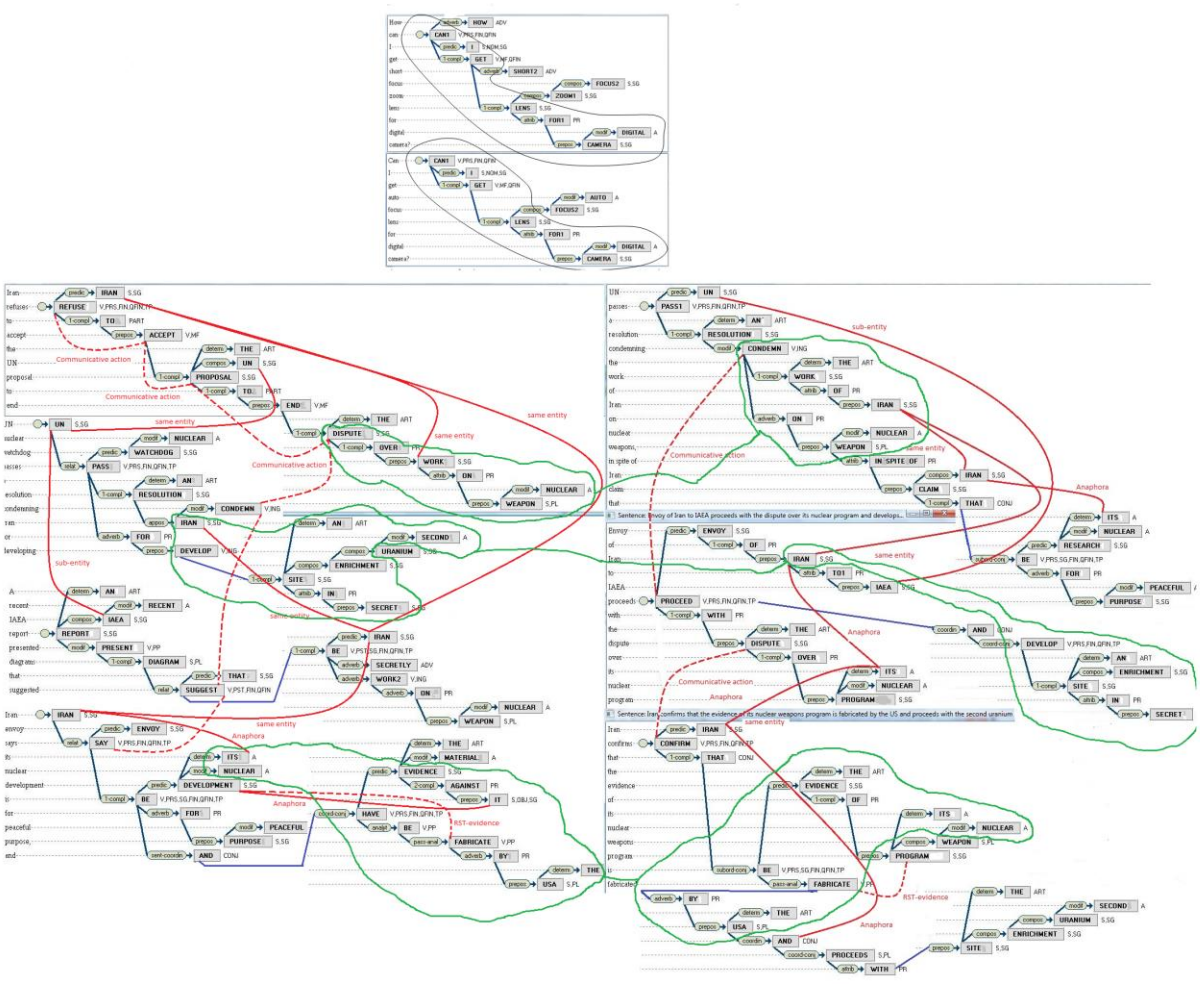

Fig.1: A high level view of the contribution of this paper: we ascend from the level of individual sentences on the top to the level of paragraphs on the bottom.

In this study we attempt to systematically extract semantic features from paragraphs of text using a graph-based learning, assuming that adequate parsing trees for individual sentences are available. In our earlier studies (Galitsky et al 2010, Galitsky et al 2012) we applied graph learning to parse trees at the sentence level, and here we proceed to learning the structure of paragraphs, relying on parse forest). 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 have applied generalizations of parse trees to the cases where a query is based on a single sentence, and candidate answers consist from single sentences (Galitsky et al 2012) and multiple sentences (Galitsky et al 2013). In these cases, to re-rank answers, we needed pair-wise

sentence-sentence generalization and sentence-paragraph generalizations respectively. In this study we rely on PT to perform a paragraph-level generalization, where both questions and answers are paragraphs of text.

The contribution of this paper is to ascend from sentence-level generalization to paragraph-level generalization (Fig.1). On the top, the common part of two parse trees is shown, which is a generalization of two sentences. On the bottom, we show a number of mapped sub-graphs which form the common parts of two PTs (graphs), which is a generalization of two paragraphs. Generalizing paragraphs of text, we do not only have to match words and their syntactic links, but their discourse relations as well. Hence overall generalization structure is much more difficult than in the case of individual sentences.

The chart for how generalization of parse thickets supports search relevance is shown in Fig. 2. The PT generalization system inputs a search query and candidate answers (obtained from local search index or using Bing search engine API). PT is computed for the query, which is a paragraph of text, and each candidate answer. Then each answer is generalized with the query to obtain the similarity score, which is the basis for relevance assessment. As a result, the answers with high similarity score is outputted as relevant.

We define the operation of generalization of text paragraphs via generalization of respective PTs to assess similarity between them. The use of generalization for similarity assessment is inspired by structured approaches to machine learning versus unstructured statistical alternatives where similarity is measured by a distance in feature space. Our intention is to extend the operation of least general generalization (e.g., the anti-unification of logical formulas) towards structural representations of paragraph of texts. Hence we will define the operation of generalization on a pair of PT as finding the maximal common sub-thickets and outline two approaches to it:

- Based on generalizing phrases from two paragraphs of text
- Based on generalizing graph representation for PTs for these paragraphs.

To represent the structure of a paragraph of text, given parse trees of its sentences, we introduce the notion of Parse Thicket (PT) as a sequence of parse trees.

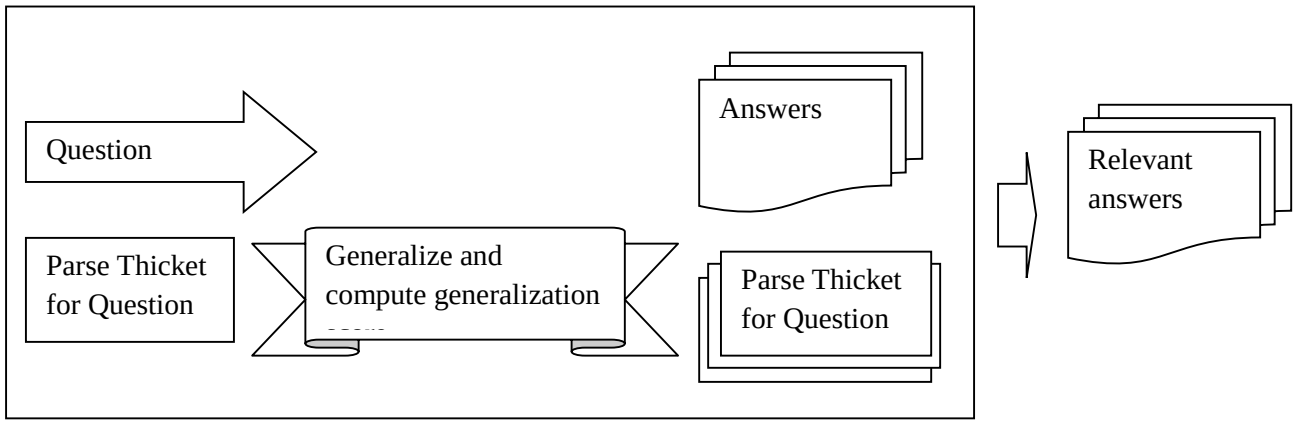

Fig. 2: Illustration for the main contribution of this paper.

2. Rhetoric Structures and Speech Acts as inter-sentence links

In this section we attempt to treat computationally, with a unified framework, two approaches to textual discourse:

- Rhetoric structure theory (RST, Mann et al 1992);
- Speech Act theory (SpActTheory, Searle 1969);

We selected these theories as most reliable and frequent sources of links between sentences. 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. For RST, we will use explicit rules which will be applied to each sentence, attempt to extract an RST relation, and add a link to the PT. For SpActT, we use a vocabulary of communicative actions to find their subjects and add respective links to PT. In this section we provide the background for these theories of discourse and explain how to use them to build links between sentence building parse thickets. In Section 4 we will show how to use these PT arcs for generalization operation.

2.1 Adapting RST for multi-sentence search

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'. 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.

When one searches for document as an answer to a question, it can reside in multiple portions in this document. To link these portions automatically, one needs to link these portions somehow, and RST relation can be leveraged. Searching for a review on "*white iPhone case*", it can read

" I got a case for my iPhone … Sentence 2 … Sentence 3 … ***But*** *when I got white case …"*

Here conjunction *But* indicates the connection between the First sentence and Fourth sentence, which happen to contain the phrase which matches *white iPhone case.* Notice that not all sentences containing keywords we seek can be matched: if there is no corresponding discourse relation, *white* can refer to another object, not *iPhone case*.

RST was originally developed as part of studies of computer-based text generation at Information Sciences Institute in 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, which is employed in building PTs. 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.

Text parsing using RST has also been attempted, although not as enthusiastically. Marcu (1997) presented an algorithm to parse the discourse structure of texts, using discourse markers as indicators of relations. (Corston and Oliver (1998) included other sources of information: whether the span in question is a main, coordinate, or subordinate clause; position of clause (main subordinate or subordinate-main); presence of certain adverbs; presence of pronouns; polarity of the clause, etc. (Le and Abeysinghe 2003) combine discourse markers, syntactic relations, and cohesive devices. Schilder (2002) uses discourse markers and position, to parse discourse structure of a slightly different form, using Segmented Discourse Representation Theory (Asher and Lascarides 2003).

We now introduce the relations of RST. Discourse units are clauses with verbs and its obligatory arguments, sentences, titles, and section titles even when they are verb-less. Both restrictive and non-restrictive modifiers that use a finite verb are embedded units. Adjacent text spans are linked together via rhetorical (coherence) relations. Mononuclear relations hold between 2 text spans, where one of them, the nucleus, is more salient to the discourse structure, and the other one, the satellite, represents supporting information. If you show them this movie - they'll scream.

Multinuclear hold between 2 or more text spans, each of which has the same weight in the discourse structure. [Go to url. Find menu item 'register'. Type your name]
Mononuclear relation links nucleus with its satellite. As an answer, satellite is only valid as long as satellite is present.

We now introduce the rules to extract RST relations between elementary discourse units in the following form:

Syntactic template

Build-link(Part-of- Syntactic template)

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

Bild-link(Part-of- Syntactic template)

is a set of phrases which will be linked. *Part-of- Syntactic template* is a set of sub-lists of *Syntactic template phrases.*

For the purpose of building PT, we construct 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 enumerate the relations and give example of some templates we use to detect an RST links, and specify respective linking rules.

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

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

---

link(NP, VP)

{*remember, recall, notice*} ∈ *ImperMentalVB.*

In-response to NP ResponseVP

---

link(NP, ResponseVP)

NP AllowVP to ResultVP

---

link(NP, ResultVP)

{*allow, help, assist, give-ability*}∈ *ResultVP*

2. Manner, Means, Medium

'Medium demonstrates … feature … of the system'

$$\frac{\textit{MeansNP DemoVB NP to ToNP}}{\textit{link(NP, ToNP)}}$$

{*show, demonstrate, indicate*} ∈ *DemoVB*

3. Temporal-before, Temporal-same-time, Temporal-after

$$\frac{\textit{VP until UntilVP}}{\textit{link(NP, UntilNP)}}$$

4. Elaboration, Elaboration-Set-Member, Elaboration-Process-Step,

Example, Elaboration-Additional

$$\frac{\textit{NP is UsedVP to AchieveGerundVP}}{\textit{link(NP, AchieveGerundVP)}}$$

{*can-be-used, should-be-employed*} ∈ *UsedVP*

Once rhetoric relation in one text is found, it can be generalized with this relation in another text. We propose a definition to do that, which will be the part of overall PT generalization:

Definition 2.1.1: RST-based generalization for texts

$text1 \wedge text2 = \cup_{i,j}\ rstRelation1_{i,}(\ldots,\ldots) \wedge rstRelation2_{i,}(\ldots,\ldots)$.

$i \in$ (*RST relations in text1*), $j \in$ (*RST relations in text2*). Notice that individual RST generalization will be defined in Section 4.1.

2.2 Adapting Speech Act Theory for multi-sentence search

In this section we formulate the Theory of Speech Act in a way applicable to finding links between sentences for paragraph-level generalization. In this theory, dialogue, negotiation, conflict dispute are forms of interactions between human agents. Elements of the language which expresses these interactions are referred to as locutions, speech acts (Bach & Harnish 1979), utterances, or *communicative actions* (we are going to keep using the last term and use abbreviation CA).

How can CAs serve as links between sentences and help answer questions? For the same question we used in the previous Section, *white iPhone case*, let us consider a document "*I asked them about cases for iPhone … I needed it to protect my phone … Sentence 2 … Sentence 3… They answered that they had a white case".* What is important here is the link between CA *asks* and CA *answered*, passing through multiple sentences. The fact that the latter CA is a reaction (response) to the former is a basis for our conclusion that the subjects of these CAs { cases for iPhone, had a white case} are correlated. Hence the phrases expressing these subjects can be linked, and matched together against a question. To determine that CA1 is a reaction to CA2, we need to establish their attributes, leveraging Speech Act Theory.

Definition 2.2.1. A communicative action is a functor of the form

verb (agent, subject, cause)

where *verb* characterizes some kind of interaction between people (e.g., explain, confirm, remind, disagree, deny, etc.), *agent* identifies the person , and subject refers to the information transmitted or object described, and cause refers to the motivation or explanation for the subject.

The foundation of the current theory of Speech Acts was developed by (Austin 1962) where he explores the performative utterances, aiming to prove that when people speak, they are doing more than simply conveying information—they act. A speech act is essentially a theory that asserts the claim that *in saying something, we perform something*. It is an action that is performed by means of language. An example would be a performative act of a judge during a hearing when he says "I now pronounce that the case is solved." Due to Austin's designation of speech acts, sentences like this adopt a notion of action. The judge's sentence is not a report of the action; it *is* the action indeed. Austin distinguishes between three types of linguistic acts: the act *of* saying something, what one *does* in saying it, and what one does *by* saying it. He labels them *Locutionary*, *Illocutionary*, and *Perlocutionary*, respectively (Farrell 2006)). A locutionary act is simply saying something about the world, e.g. a declarative sentence such as "The product does not work." This sentence is not posing a question, promising, or commanding anything. It simply states something about the world, containing purely propositional content. This type of act is the most basic, and does not require much more explanation.

The illocutionary act includes promising, questioning, admitting, hypothesizing, etc. The locutionary act was simply the act of saying something, while the illocutionary act is performed *in* saying something. For example, "A company promises to support the product after it is sold" asserts more than simply stating a sentence about the world. It includes an *assertion* that is performative in nature. Illocutionary acts are very prominent in language, and are frequently in use in complaint scenarios. The third type of linguistic acts are perlocutionary ones. These are non-conventional sentences that cause a natural condition or state in a person. These acts de-emphasize the actual intentions, and focus on the effects on the hearer. For example, acts of frightening or convincing depend on the response of another person. If a perlocutionary act is successful, then one can state that an illocutionary acts has successfully taken place.

Approximating scenarios of multiagent interactions, we follow along the lines of communicative actions' division into *constatives* and *performatives*.

- Constatives describe or report some state of affairs such that it is possible to assess whether they are false or true.
- Performatives, on the other hand, are *fortunate* or *unfortunate, sincere* or *insincere*, *realistic* or *unrealistic*, and, finally valid or invalid, which is the focus of the current study. Performatives address the attitude of an the agent performing the linguistic act, including his thoughts, feelings, and intentions.

To choose communicative actions to adequately represent an interaction between humans, we have selected the most frequently used ones from our structured database of complaints (Table 1, Galitsky et al 2009).

Table 1**.** The set of communicative actions

A person describes his own action	A person describes an opponent's action
Agree, explain, suggest, bring company's attention, remind, allow, try, request, understand, inform, confirm ask, check, ignore, convince	Agree, explain, suggest, remind, allow, try, request, understand, inform, confirm, ask, check, ignore, convince, disagree, appeal, deny, threaten, bring to customer's attention, accept complaint, accept /deny responsibilities, encourage, cheat

disagree, appeal, deny, threaten	

A number of computational approaches have attempted to discover and categorize how the agents' attitudes and communicative actions are related to each other in the case of computational simulation of human agents (Searle 1969; Cohen & Levesque 1990). Applying machine learning to the attitudes and communicative actions, we are primarily concerned with how these approaches can provide a unified and robust framework for computing similarity between the communicative actions. The theory of speech acts seems to be one of the most promising approaches to categorizing communicative actions in terms of their roles. Following (Bach & Harnish 1979), we consider four categories of illocutionary communicative actions with major representatives *stating, requesting, promising and apologizing.* Each speech act is related to a single category only in the framework of the speech act theory.

To extend the speech act–based means of expressing similarity between communicative actions, we introduce five attributes each of which reflects a particular semantic parameter for communicative activity:

- *Positive/ negative attitude* expresses whether a communicative action is a cooperative (friendly, helpful) move (1), uncooperative (unfriendly, unhelpful) move (-1), neither or both (hard to tell, 0).
- *Request / respond mode* specifies whether a communicative action is expected to be followed by a reaction (1), constitutes a response (follows) a previous request, neither or both (hard to tell, 0).
- *Info supply / no info supply* tells if a communicative action brings in an additional data about the conflict (1), does not bring any information (-1), 0; does not occur here.
- *High / low confidence* specifies the confidence of the preceding mental state so that a particular communicative action is chosen, high knowledge/confidence (1), lack of knowledge/confidence (-1), neither or both is possible (0).
- *Intense / relaxed mode* says about the potential emotional load: high (1), low (-1), neutral (0) emotional loads are possible.

In order to define a robust framework to identify CA arcs in PT we are interested in distinguishing which are the most common communicative actions used in complaint scenarios, and how they can be clustered in terms of the attitudes commonly associated with them. Table 2 shows the attribute of CA well suited for generalization. For example, *agree^accept* = <1, -1, -1, 1, *>

Table 2: Extended attributes of communicative actions

Communicative action	**Attributes**				
	Positive/ negative attitude	**Request / respond mode**	**Info supply / no info supply**	**High / low confidence**	**Intense / relaxed mode**
agree	1	-1	-1	1	-1
accept	1	-1	-1	1	1
explain	0	-1	1	1	-1
suggest	1	0	1	-1	-1
bring_attention	1	1	1	1	1
remind	-1	0	1	1	1
allow	1	-1	-1	-1	-1
try	1	0	-1	-1	-1
request	0	1	-1	1	1

understand	0	-1	-1	1	-1
Inform	0	0	1	1	-1
confirm	0	-1	1	1	1
ask	0	1	-1	-1	-1
check	-1	1	-1	-1	1
ignore	-1	-1	-1	-1	1
convince	0	1	1	1	-1
disagree	-1	-1	-1	1	-1
appeal	-1	1	1	1	1
deny	-1	-1	-1	1	1
threaten	-1	1	-1	1	1

Having built the similarity model for communicative actions, we can now compute word-word similarity for them via attributes.

3. Parse thickets and their graph representation

3.1 Generalizing portions of text

To measure the similarity of abstract entities expressed by logic formulas, a least-general generalization (Plotkin 1970) was proposed for a number of machine learning approaches, including explanation-based learning and inductive logic programming. Given two logical terms, it produces a more general term that covers both.

Definition 3.1.1: Let E_1 and E_2 be two terms. Term E is a *generalization* of E_1 and E_2 if there exist two substitutions σ_1 and σ_2 such that $\sigma_1(E) = E_1$ and $\sigma_2(E) = E_2$. The most specific generalization of $E1$ and $E2$ is called the anti-unifier. Here, we apply this abstraction to anti-unify such data as texts that are traditionally referred to as unstructured.

In this study, to measure the similarity between portions of text such as paragraphs, sentences and phrases, we extend the notion of generalization from logic formulas to the sets of syntactic parse trees of these portions of text. If it were possible to define the similarities between natural language expressions at a purely semantic level, the least-general generalization would be sufficient. However, in horizontal search domains where the construction of full ontologies for a complete translation from natural language to logic language is not plausible, an extension of the abstract operation of generalization to the syntactic level is required. Rather than extracting common keywords, the generalization operation produces a syntactic expression that can be semantically interpreted as a common meaning shared by two sentences.

Let us represent a meaning of two NL expressions using logic formulas and then construct the unification and anti-unification of these formulas. How can we express a commonality between the expressions?

- *camera with digital zoom*
- *camera with zoom for beginners*

To express the meanings, we use the predicates *camera(name_of_feature, type_of_users)* (in real life, we would have a much higher number of arguments), and zoom(*type_of_zoom).* The above NL expressions will be represented as follows:

camera(zoom(digital), AnyUser)

camera(zoom(AnyZoom), beginner),

where the variables (uninstantiated values that are not specified in NL expressions) are capitalized. Given the above pair of formulas, unification computes their most general specialization *camera(zoom(digital), beginner),* and anti-unification computes their most specific generalization, *camera(zoom(AnyZoom), AnyUser).*

At the syntactic level, we have the generalization of two noun phrases as follows:

{NN-camera, PRP-with, [digital], NN-zoom [for beginners]}.

We eliminate the expressions in square brackets because they occur in one expression and do not occur in another. Thus, we obtain

{NN-camera, PRP-with, NN-zoom]}, which is a syntactic analog to the semantic generalization above.

The purpose of an abstract generalization is to find the commonality between portions of text at various levels. The generalization operation occurs on the following levels:

- Article
- Paragraph
- Sentence
- Phrases (noun, verb and others)
- 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 pair-wise generalization of these expressions.

Definition 3.1.2: 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 3.1.3: 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 3.1.2.
- 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 (Lehmann 1982).

Definition 3.1.4: 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 as a common meaning shared by two sentences.

Input: A pair of sentences Output: A list of lists of generalized phrases for each phrase type

1) Obtain the parsing tree for each sentence. For each word (tree node), we have a lemma, a part of speech and the form of the word's information. This information is contained in the node label. We also have an arc to the other node.
2) Split sentences into sub-trees that are phrases for each type: verb, noun, prepositional and others. These sub-trees are overlapping. The sub-trees are coded so that the information about their occurrence in the full tree is retained.
3) All the sub-trees are grouped by phrase types.
4) Extend the list of phrases by adding equivalence transformations (Section 2.1.1).
5) Generalize each pair of sub-trees for both sentences for each phrase type.
6) For each pair of sub-trees, yield an alignment (Gildea 2003), and generalize each node for this alignment. Calculate the score for the obtained set of trees (generalization results).
7) For each pair of sub-trees of phrases, select the set of generalizations with the highest score (the least general).
8) Form the sets of generalizations for each phrase type whose elements are the sets of generalizations for that type.
9) Filter the list of generalization results: for the list of generalizations for each phrase type, exclude more general elements from the lists of generalization for a given pair of phrases.

For a given pair of words, only a single generalization exists; if words are the same in the same form, the result is a node with this word in this form. We refer to the generalization of words occurring in a syntactic tree as a *word node*. If the word forms are different (e.g., one is single and the other is plural), only the lemma of the word remains. If the words are different and only the parts of speech are the same, the resultant node contains only the part-of-speech information with no lemma. If the parts of speech are different, the generalization node is empty.

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

To buy the digital camera today, on Monday

The digital camera was a good buy today, the first Monday of the month

The generalization is {<*JJ-digital, NN-camera*>, <*NN-today, ADV,Monday*>}, where the generalization for the noun phrase is followed by the generalization for the adverbial phrase. The 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 a different sequence in the other generalization nodes.

We can see that the multiple maximum generalizations occur depending on how the correspondence between words is established; multiple generalizations are possible. To obey the condition of the maximum, we introduce a *score* for generalization. The scoring weights of generalizations are decreasing,

roughly, in the following order: nouns and verbs, other parts of speech, and nodes with no lemma, only a part of speech. In its style, the generalization operation follows the notion of the 'least-general generalization' or anti-unification, if a node is a formula in a language of logic. Therefore, we can refer to the syntactic tree generalization as an operation of *anti-unification of syntactic trees*.

To optimize the calculation of the generalization score, we conducted a computational study to determine the POS weights and deliver the most accurate similarity measure possible between sentences (Galitsky et al 2010, Galitsky et al 2012). The problem was formulated as finding the optimal weights for nouns, adjectives, verbs and their forms (such as gerund and past tense) so that the resultant search relevance is maximized. The search relevance was measured as a deviation in the order of search results from the best result for a given query (delivered by Google). The current search order was determined based on the score of generalization for the given set of POS weights (the other generalization parameters having been fixed). As a result of this optimization, we obtained $W_{NN} = 1.0$, $W_{JJ} = 0.32$, $W_{RB} = 0.71$, $W_{CD} = 0.64$, $W_{VB} = 0.83$, and $W_{PRP} = 0.35$, excluding common and frequent verbs like *get/ take/set/put*, for which $W_{VBcommon} = 0.57$. We also establish that $W_{<POS,*>} = 0.2$ (different words but the same POS), and $W_{<*,word>} = 0.3$ (the same word occurring as different POSs in two sentences).

Definition 3.1.1 The generalization score (or the similarity between the sentences $sent_1$ and $sent_2$) can then be expressed as a sum through the phrases of the weighted sum for the words $word_{sent1}$ and $word_{sent2}$

$score(sent_1, sent_2) =$

$$\sum_{\{NP, VP, \ldots\}} \sum W_{POS}\ word\ generalization(word_{sent1}\ word_{sent2}).$$

The (maximal) generalization can then be defined as the generalization with the highest score. Accordingly, we define the generalization for phrases, sentences and paragraphs. The result of the generalization can be further generalized with other parse trees or generalizations. For a set of sentences, the totality of the generalizations forms a lattice: the order of the generalizations is set by the subsumption relation and the generalization score. We enforce the associativity of the generalization of parse trees by means of computation: it must be verified and the resultant list must be extended each time a new sentence is added. Note that such an associativity is not implied in our definition of generalization.

3.1.1 Equivalence transformation of phrases

We have manually created and collected a rule base for generic linguistic phenomena from various resources. Unlike a text entailment system, in our situation we do not require a complete transformation system as long as we have a sufficiently rich set of examples. Syntactic-based rules capture the entailment inferences associated with common syntactic structures, including the simplification of the original parse tree, reducing it into a canonical form, extracting its embedded propositions, and inferring propositions from the non-propositional sub-trees of the source tree (Table 3).

Category	Original/*Transformed fragment*
Conjunctions	Camera is very stable and *has played an important role in filming their wedding*
Clausal modifiers	The flash was disconnected when the *children went out to play in the yard*
Relative clauses	I was forced to close the *LCD*, which *was blinded by the sun*
Appositives	*Digital zoom, a feature provided by the new generation of cameras*, dramatically decreases the images' sharpness
Determiners	My customers use their (*an* auto …) autofocus camera for polar expeditions (their => *an*)
Passive	A cell phone can be easily grasped in the palm of a hand (*The hand can easily grasp the cell phone in its palm*)
genitive modifier	Sony's LCD screens work as well as Canon's in a sunny environment (*The LCD of Sony… as well as that of Canon*)
Polarity	It made me use digital zoom for mountain shots (I *used* digital zoom…)

Table 3: Rules of graph reduction for generic linguistic structures. The resultant reductions are italicized.

The valid matching of sentence parts embedded as verb complements depends on the verbs' properties and the polarity of the context in which the verb appears (positive, negative, or unknown). We used the list of verbs for communicative actions in (Galitsky and Kuznetsov 2008), which indicate positive polarity context. This list is complemented with a few reporting verbs, such as *say* and *announce*, because opinions are often provided in reported speech in the news domain, while authors are usually considered reliable. We also used annotation rules to mark the negation and modality of predicates (mainly verbs) based on their descendent modifiers.

An important class of transformation rules involves noun phrases. The adjectives of a single noun group can be re-sorted, as well as all nouns except the head one. A noun phrase that is a post-modifier of the head noun of a given phrase can be merged with the latter. The resultant meaning may be distorted; otherwise, we would miss important commonalities between expressions containing noun phrases. For an expression 'NP_1 <*of* or *for*> NP_2', we form a single NP with the head noun *head*(NP_2), *head*(NP_1) playing the role of modifier, and an arbitrary sorting of adjectives. For example, we convert '*camera with digital zoom'* into '*digital zoom camera', head(NP) = camera*.

3.2 Finding similarity between two paragraphs of text

To rank search results, one needs to have a measure of similarity between questions and answers. Once a candidate set of answers is obtained by a keyword search (using, for example TF*IDF model), we calculate

the similarity between the question and each of its candidate answers, and rank the latter set respectively, in the order of similarity decrease.

We will compare three following approaches to assessing the similarity of text paragraphs:

- Baseline: bag-of-words approach, which computes the set of common keywords/n-grams and their frequencies.
- Pair-wise sentence matching: we will apply syntactic generalization to each pair of sentences, and sum up the resultant commonalities. This technique has been developed in our previous work (Galitsky et al 2012, Section 2.1).
- Paragraph-paragraph match.

The first approach is most typical for industrial NLP applications today, and the second one has been explored in our previous studies. Kernel-based approach to parse tree similarities (Moschitti 2006, Zhang et al 2008), as well as tree sequence kernel (Sun et al 2011), being tuned to parse trees of individual sentences, also belong to the second approach. We demonstrate its richness of the third approach here, and in the consecutive sections we will provide a step-by-step illustration for how PTs are constructed and generalized. We will introduce a pair of short texts (articles) and compare the above three approaches. This example will go through the whole section.

The first paragraph can be viewed as a search query, and the second paragraph can be viewed as a candidate answer. A relevant answer should be a closely related text on one hand, and not a piece of duplicate information on the other hand.

Operator '^' in the following example and through all the paper denotes *generalization* operation. Describing parse trees we use standard notation for constituency trees: **[…]** represents sub-phrase, NN, JJ, NP etc. denote parts-of-speech and types of sub-phrases, '*****' will be is a placeholder for a word, part of speech, or an arbitrary graph node.

"Iran refuses to accept the UN proposal to end the dispute over work on nuclear weapons",

"UN nuclear watchdog passes a resolution condemning Iran for developing a second uranium enrichment site in secret",

"A recent IAEA report presented diagrams that suggested Iran was secretly working on nuclear weapons",

"Iran envoy says its nuclear development is for peaceful purpose, and the material evidence against it has been fabricated by the US",

^

"UN passes a resolution condemning the work of Iran on nuclear weapons, in spite of Iran claims that its nuclear research is for peaceful purpose",

"Envoy of Iran to IAEA proceeds with the dispute over its nuclear program and develops an enrichment site in secret",

"Iran confirms that the evidence of its nuclear weapons program is fabricated by the US and proceeds with the second uranium enrichment site"

The list of common keywords gives a hint that both documents are on nuclear program of Iran, however it is hard to get more specific details

Iran, UN, proposal, dispute, nuclear, weapons, passes, resolution, developing, enrichment, site, secret, condemning, second, uranium

Pair-wise generalization gives a more accurate account on what is common between these texts:

[NN-work IN-* IN-on JJ-nuclear NNS-weapons], [DT-the NN-dispute IN-over JJ-nuclear NNS-*], [VBZ-passes DT-a NN-resolution],

[VBG-condemning NNP-iran IN-*],

[VBG-developing DT-* NN-enrichment NN-site IN-in NN-secret]],

[DT-* JJ-second NN-uranium NN-enrichment NN-site]],

[VBZ-is IN-for JJ-peaceful NN-purpose],

[DT-the NN-evidence IN-* PRP-it], [VBN-* VBN-fabricated IN-by DT-the NNP-us]

Parse Thicket generalization gives the detailed similarity picture which looks more complete than the pair-wise sentence generalization result above:

[NN-Iran VBG-developing DT-* NN-enrichment NN-site IN-in NN-secret]

[NN-generalization-<UN/nuclear watchdog> * VB-pass NN-resolution VBG condemning NN- Iran]

[NN-generalization-<Iran/envoy of Iran> Communicative_action DT-the NN-dispute IN-over JJ-nuclear NNS-*

[Communicative_action - NN-work IN-of NN-Iran IN-on JJ-nuclear NNS-weapons]

[NN-generalization <Iran/envoy to UN> Communicative_action NN-Iran NN-nuclear NN-* VBZ-is IN-for JJ-peaceful NN-purpose],

Communicative_action - NN-generalize <work/develop> IN-of NN-Iran IN-on JJ-nuclear NNS-weapons]*

[NN-generalization <Iran/envoy to UN> Communicative_action NN-evidence IN-against NN Iran NN-nuclear VBN-fabricated IN-by DT-the NNP-us]

condemn^proceed [enrichment site] <leads to> suggest^condemn [work Iran nuclear weapon]

One can feel that PT-based generalization closely approaches human performance in terms of finding similarities between texts. To obtain these results, we need to be capable of maintaining coreferences, apply the relationships between entities to our analysis *(subject vs relation-to-this subject),* including relationships between verbs *(develop* is a partial case of *work).* We also need to be able to identify communicative actions and generalize them together with their subjects according to the specific patterns of Speech Act Theory. Moreover, we need to maintain Rhetoric Structure relationship between sentences, to generalize at the level of textual discourse.

Fig. 3 and Fig. 4 show the dependency-based parse trees for the above texts T1 and T2. Each tree node has labels as part-of-speech and its form (such as SG for ‘single’); also, tree edges are labeled with the syntactic connection type (such as ‘composite’). We are not concerned with a particular type of a parse tree representation (dependency or constituency), as long as we have labels for nodes and vertexes.

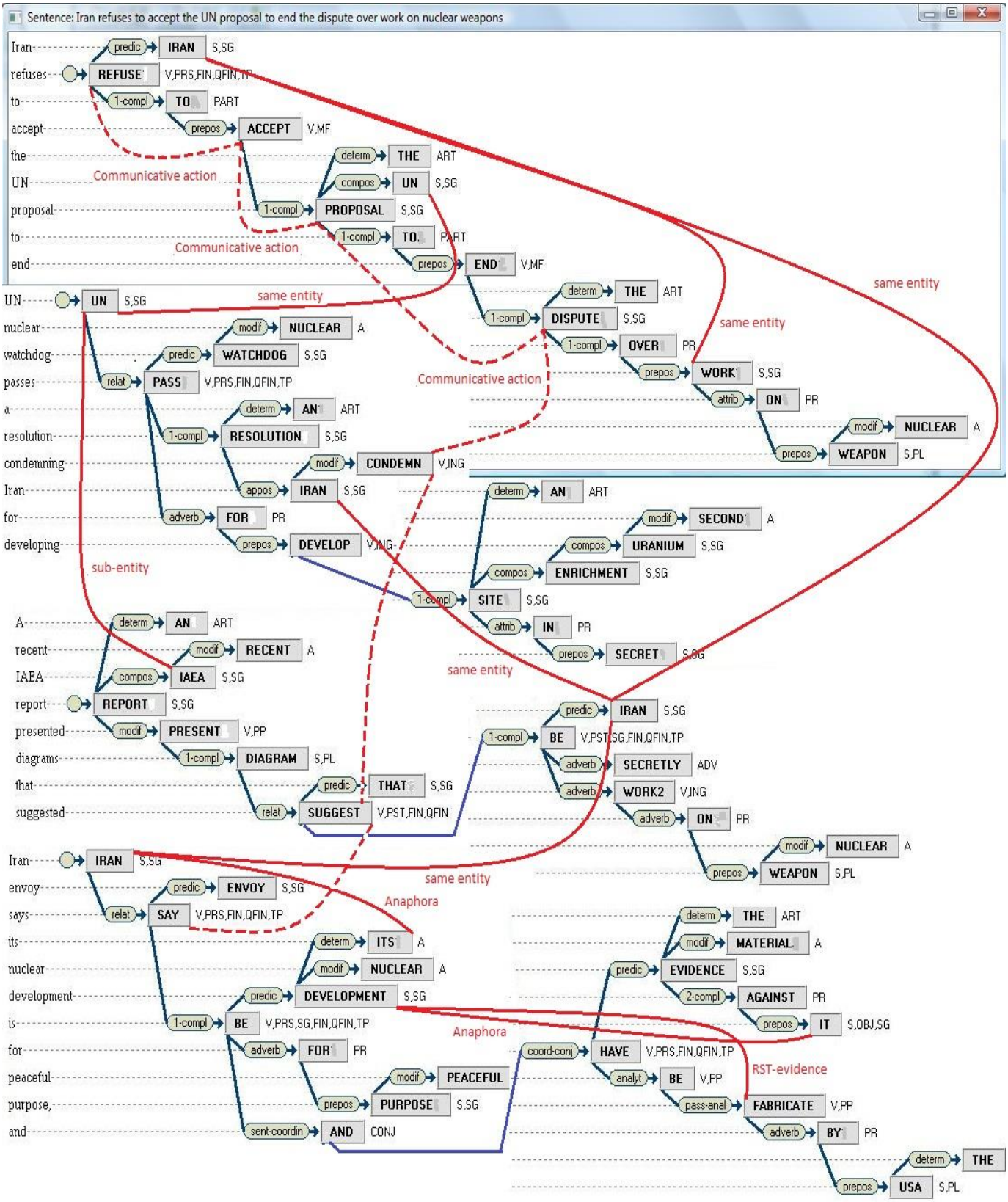

Fig. 3: Parse thicket for text T1.

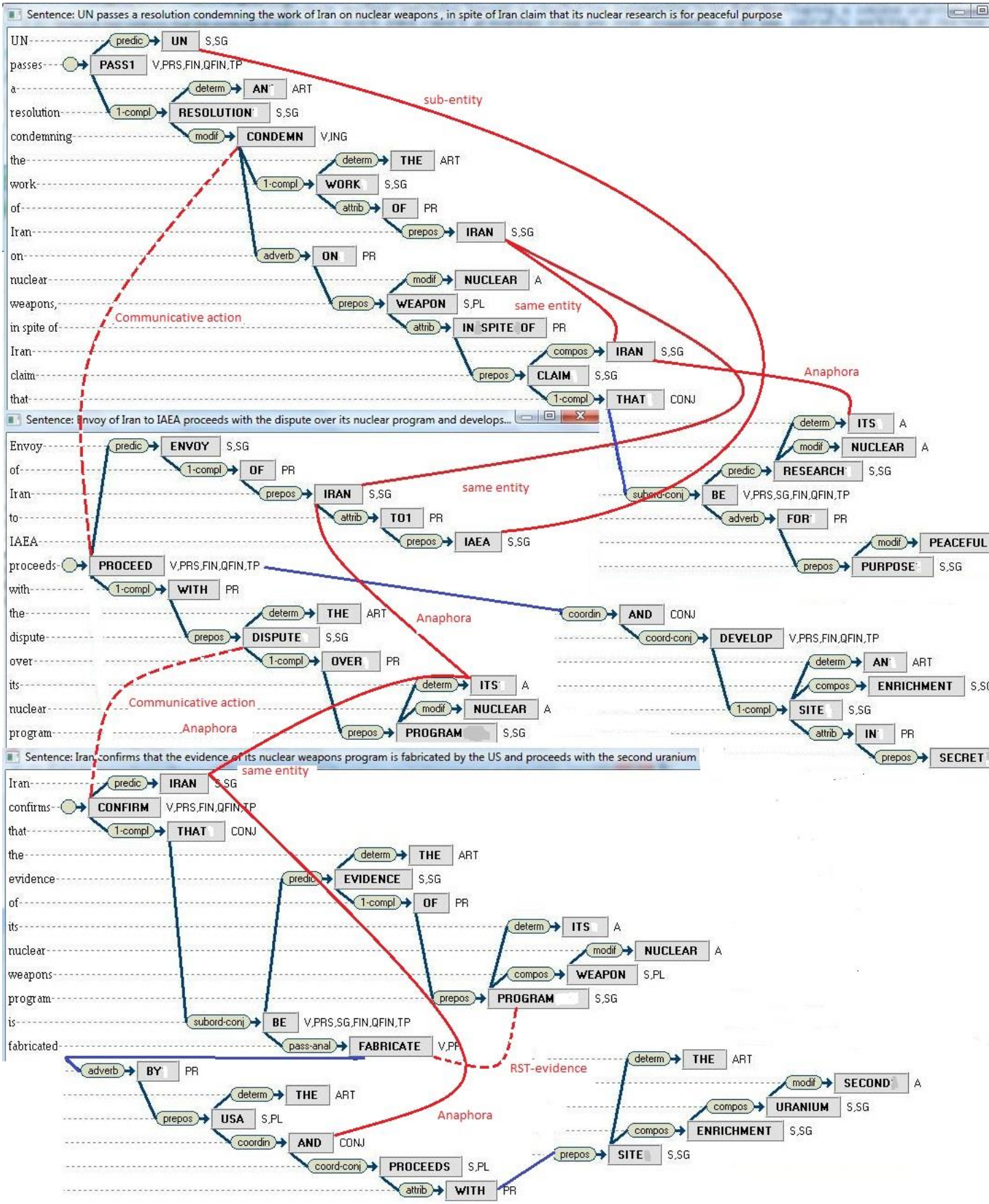

Fig. 4: Parse thicket for text T2.

3.3 How coreferences help search recall

We now proceed to an example of search query and two answers, where coreference establishes a link between two sentences and makes the linked words relevant to a query. We will consider two cases for text indexing, where establishing proper coreferences inside and between sentences connects entities in an index for proper match with a question:

Question: Which specialist doctor should treat my tuberculosis?

Text for indexing/candidate answer 1: … Tuberculosis is usually a lung disease. It is cured by doctors specializing in pulmonology.

Text for indexing/candidate answer 2: … Tuberculosis is a lung disease… Pulmonology specialist Jones was awarded a prize for curing a special form of disease.

In the first case, establishing the 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. One can observer that building PT for an answer online and matching it with question delivers answers which would be determined as irrelevant without building coreference chains. This way coreferences and same entity relations as parts of PTs improve search recall. From now on we will refer to such chains as phrases from distinct sentences as *thicket phrases*.

3.4 How rhetoric relation improve search accuracy

We introduce a domain where a pair-wise comparison of sentences is insufficient to properly learn certain semantic features of texts. This is due to the variability of ways information can be communicated in multiple sentences, and variations in possible discourse structures of text which needs to be takes into account.

We consider an example of text classification problem, where short portions of text belong to two classes:

- Tax liability of a landlord renting office to a business.
- Tax liability of a business owner renting an office.

I rent an office space. This office is for my business. I can deduct office rental expense from my business profit to calculate net income.

To run my business, I have to rent an office. The net business profit is calculated as follows. Rental expense needs to be subtracted from revenue.

To store goods for my retail business I rent some space. When I calculate the net income, I take revenue and subtract business expenses such as office rent.

I rent out a first floor unit of my house to a travel business. I need to add the rental income to my profit. However, when I repair my house, I can deduct the repair expense from my rental income.

I receive rental income from my office. I have to claim it as a profit in my tax forms. I need to add my rental income to my profits, but subtract rental expenses such as repair from it.

I advertised my property as a business rental. Advertisement and repair expenses can be subtracted from the rental income. Remaining rental income needs to be added to my profit needs to be reported as taxable profit.

Firstly, note that keyword-based analysis does not help to separate the first three paragraph and the second three paragraphs. They all share the same keywords *rental/office/income/profit/add/subtract.* Phrase-based analysis does not help, since both sets of paragraphs share similar phrases

Secondly, pair-wise sentence comparison does not solve the problem either.

Anaphora resolution is helpful but insufficient. All these sentences include 'I' and its mention, but other links between words or phrases in different sentences needs to be used.

Rhetoric structures need to come into play to provide additional links between sentences. The structure to distinguish between

renting for yourself and deducting from total income and

renting to someone and adding to income embraces multiple sentences. The second clause about adding/subtracting incomes is linked by means of the rhetoric relation of *elaboration* with the first clause for landlord/tenant. This rhetoric relation may link discourse units within a sentence, between consecutive sentences and even between first and third sentence in a paragraph. Other rhetoric relations can play similar role for forming essential links for text classification.

3.5 Thicket Phrases and their generalization

Once we have a sequence of parse trees for a question, and that of an answer, how can we match these sequences? A number of studies compute pair-wise similarity between parse trees (Collins and Duffy, 2002; Punyakanok *et al.*, 2003; Moschitti, 2006). However, to rely upon discourse structure of paragraphs, and to avoid dependence on how content is distributed through sentences, we represent the whole paragraphs of questions and answers as a single graph. To determine how good is an answer for a question, we match their respective PTs and assign a score to the size of common sub-PT.

We extend the syntactic relations between the nodes of the syntactic dependency parse trees towards more general text discourse relations. Once we have such relations as "the same entity", "sub-entity", "super-entity" and anaphora, we can extend the notion of phrase to be matched between texts. In case of single sentences, we match noun, verb, and other types of phrases in questions and answers. In case of multiple sentences in each, we extend the notion of phrases so that they are independent of how information being communicated is split into sentences. Relations between the nodes of parse trees (which are other than syntactic) can merge phrases from different sentences or from a single sentence, which are not syntactically connected. We will refer to such extended phrases as thicket phrases.

If words *X* and *Y* are connected by a coreference relation, an index needs to include the chain of words $X_0, X_1 \ldots X, Y_0, Y_1 \ldots Y$, where chains $X_0, X_1 \ldots X$ and $Y_0, Y_1 \ldots Y$ are already indexed (phrases including *X* and *Y*). Hence establishing coreferences is important to extend index in a way to improve search recall. Usually, keywords from different sentences can only be matched with query keywords with a low score (high score is delivered by inter-sentence match).

Definition 3.5.1: Thicket phrases for a pair of parse trees.

If we have two parse trees P_1 and P_2 of text T_1, and an arc for a relation $r: P_{1i} \rightarrow P_{2j}$ between the nodes P_{1i} and P_{2j}, we can now match $\ldots, P_{1i-2}, P_{1i-1}, P_{1i}, P_{2j}, P_{2j+1}, P_{2j+2}, \ldots$ of T_1 against a phrase of a single sentence or a merged phrases of multiple sentences from T_2.

An example of building a thicket phrase by linking two trees is shown in Fig. 5. A thicket phrase can be thought as beginning from a phrase in one sentence, then jumping to the tree for another sentence and continuing the phrase.

Although the generalization is defined as the set of maximal common sub-graphs, its computation is based on matching phrases. To generalize a pair of sentences, we perform chunking and extract all noun, verb, prepositional and other types of phrases from each sentence. Then we perform generalization for each type of phrases, attempting to find a maximal common sub-phrase for each pair of phrases of the same type. The resultant phrase-level generalization can then be interpreted as a set of paths in resultant common sub-trees (Galitsky *et al.*, 2012).

Generalization of parse thickets, being a maximal common sub-parse thicket, can be computed at the level of phrases as well, as a structure containing maximal common sub-phrases. However, the notion of phrases is extended now: thicket phrases can contain regular phrases from different sentences. The way these phrases are extracted and formed depends on the source of non-syntactic link between words in different sentences: thicket phrases are formed in a different way for communicative actions and RST relations. Notice that the set of regular phrases for a parse thicket is a sub-set of the set of thicket phrases (all phrases extracted for generalization). Because of this richer set of phrases for generalization, the parse thicket generalization is richer than the pair-wise thicket generalization, and can better tackle variety in phrasings and writing styles, as well as distribution of information through sentences.

We will now outline the algorithm of forming thicket phrases. Most categories of thicket arcs will be illustrated below.

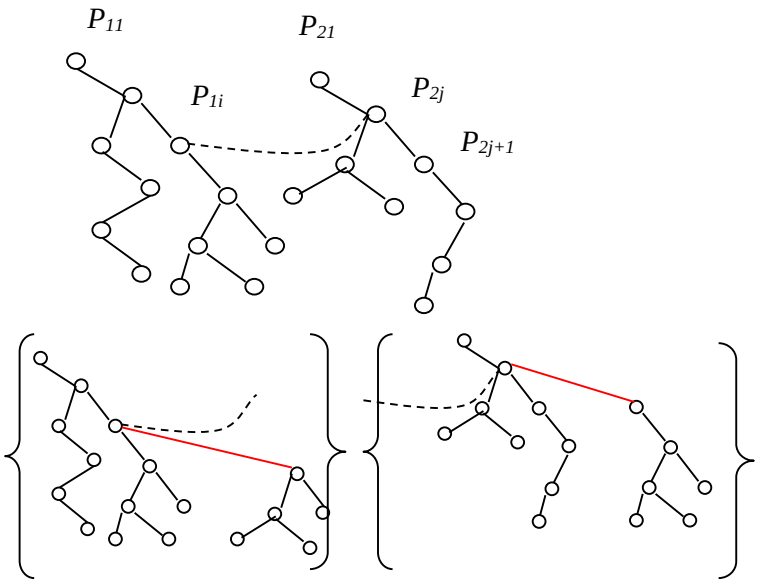

Fig. 5: An arc which connects two parse trees for two sentences in a text (on the top) and the derived set of extended trees (on the bottom).

Input: Sentences of a paragraph and their parse

For each sentence *S* in a paragraph *P*
Form a list of previous sentences in a paragraph S_{prev}
For each word in the current sentence:
- If this word is a *pronoun*: find all nouns or noun phrases in the S_{prev} which are
  * The same entities (via coreference resolution)
- If this word is a *noun*: find all nouns or noun phrases in the S_{prev} which are
  * The same entities (via coreference resolution)
  * Synonymous entity
  * Super entities
  * Sub and sibling entities
- If this word is a *verb*:
  * If it is a communicative action:
    Form the phrase for its subject $VBCA_{phrase}$, including its verb phrase Vph
    Find a preceding communicative action $VBCA_{phrase0}$ from S_{prev} with its subject and form a thicket phrase [$VBCA_{phrase}$, $VBCA_{phrase0}$]
  * If it indicates RST relation
    Form the phrase for the pair of phrases which are the subjects [$VBRST_{phrase1}$, $VBRST_{phrase2}$], of this RST relation, $VBRST_{phrase1}$ belongs to S_{prev}.

3.6 Example of Parse Thicket

Figs. 1 and 2 show examples of Parse Thickets for the above paragraphs. In addition to syntactic links between words in a sentence, words in a paragraph are connected with links of the different nature. To form a complete formal representation of a paragraph, we attempt to express as many links as possible. The most obvious links are the same entities, which is a partial case of coreferences. A less trivial tasks is to identify a sub-entity relation; an external resource needs to be consulted. In Fig. 6 we have a sub-entity link IAEA → UN. The fact that the former is a sub-entity (in this case, sub-organization) is either obtained by using lookups available as a part of general-purpose NLP system like OpenNLP, Stanford NLP (Lee at al 2012), GATE, LingPipe and others, or using web mining of sites like Wikipedia, Freebase and others. In more complex cases, such as multi-words, more complex web mining settings are required, such as (Velásquez et al 2011, Galitsky et al 2011).

The communicative links reflect the discourse structure associated with participation (or mentioning) of more than single agent in text. The links form a sequence connecting the words for communicative actions (either verbs or multi-words implicitly indicating a communicative intent of a person). We have thoroughly investigated the structure of how communicative actions occur in text in our study of argumentation in customer complaints, which is one of the most complicated cases of communicative actions-based discourse (Galitsky et al 2009).

The main observation concerning communicative actions in relation to finding text similarity is that their subjects need to be generalized in the context of these actions, and that they should not be generalized with other, "physical" actions. Hence we generalize the individual occurrences of communicative actions together with their subjects, as well as their pairs as discourse "steps". Generalization of communicative actions can also be thought from the standpoint if matching the verb frames, such as VerbNet (Palmer 2009). For a communicative action, we distinguish an actor, one or more agents being acted upon, and the phrase describing the features of this action. In the next section we will illustrate how respective arguments of verbs are generalized.

Notice the three categories of the formed thicket phrases:

- Regular phrases;
- Thicket phrases;
- SpActT phrases;
- CA phrases.

Once we collected the thicket phrases for texts T1 and T2, we can do the generalization. When we generalize thicket phrases from various categories, the following constraints should be taken into account (Table 4).

Table 4: Which phrase types can be generalized with each other

	Regular phrases	Entity-based thicket phrases	RST-based thicket phrases	SpActT-based thicket phrases
Regular phrases	Obeying phrase type +	+	+	+
Entity-based thicket phrases	+	+	-	-
RST-based thicket phrases			+	-
SpActT-based thicket phrases				+

For example, entity-based thicket phrase can be generalized with regular phrases, but with neither RST nor SpActT.

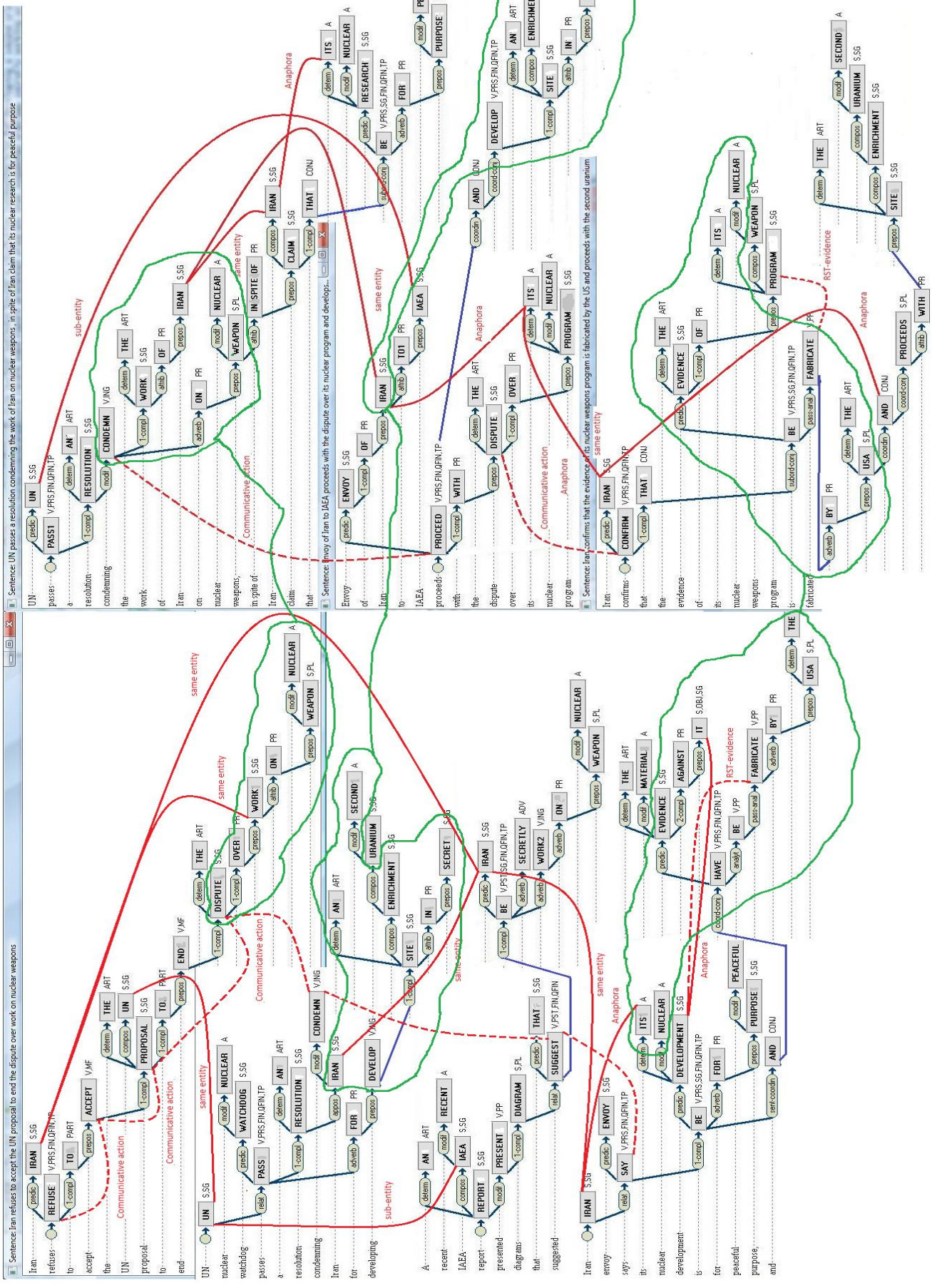

Fig. 6: Three instances of common sub-PTs are shown as connected clouds

4. Generalization of Parse Thickets

In Section 3 we defined and showed how to construct PTs. We also introduced generalization of individual parse trees. Based on that, in this section we introduce generalization of PTs which is based on generalization of individual parse trees on one hand and on generalization of discourse structures on the other hand. For coreferences and entity-entity type of relation, we link them and consider the nodes in PT identical. For RST, we attempt to extract an RST relation, and form a thicket phrase around it, including a placeholder for RST relation itself (Galitsky et al 2013). For SpActT, we use a vocabulary of communicative actions to find their subjects (Galitsky & Kuznetsov 2008), add respective arcs to PT, and form the respective sequence of thicket phrases.

4.1 Generalization for RST arcs

Two connected clouds on the right of Fig.6 show the generalization instance based on RST relation "RCT-evidence". This relation occurs between the phrases

evidence-for-what [Iran's nuclear weapon program] and what-happens-with-evidence [Fabricated by USA] on the right-bottom, and

evidence-for-what [against Iran's nuclear development] and what-happens-with-evidence [Fabricated by the USA] on the right-top.

Notice that in the latter case we need to merge (perform anaphora substitution) the phrase ' *its nuclear development'* with *'evidence against it'* to obtain *'evidence against its nuclear development'.* Notice the arc *it - development,* according to which this anaphora substitution occurred. *Evidence* is removed from the phrase because it is the indicator of RST relation, and we form the subject of this relation to match. Furthermore, we need another anaphora substitution *its- Iran* to obtain the final phrase.

As a result of generalizations of two RST relations of the same sort (evidence) we obtain

Iran nuclear NNP – RST-evidence – *fabricate by USA.*

Notice that we could not obtain this similarity expression by using sentence-level generalization.

Green clouds at indicate the sub-PTs of T_1 and T_2 which are matched. We show three instances of PT generalization.

Definition 4.1.1: $RST_{type1}(phrase_1) \wedge RST_{type2}(phrase_2) = RST_{type1} \cup phrase_1 \wedge phrase_2$

when $type_1 = type_2$, otherwise $\varnothing$. Notice that the relation itself is retained to be further generalized if required.

4.2 Generalization for CA arcs

Communicative actions are used by text authors to indicate a structure of a dialogue or a conflict (Searle 1969). Hence analyzing the communicative actions' arcs of PT, one can find implicit similarities between texts. We can generalize:

1. one communicative actions from with its subject from T1 against another communicative action with its subject from T2 (communicative action arc is not used) ;
2. a pair of communicative actions with their subjects from T1 against another pair of communicative actions from T2 (communicative action arcs are used) .

In our example, we have the same communicative actions with subjects with low similarity:

condemn [‘Iran for developing second enrichment site in secret’] vs *condemn* [‘the work of Iran on nuclear weapon’] , or different communicative actions with similar subjects.

Looking on the left (bottom) of Fig. 6 one can observe two connected clouds: the two distinct communicative actions *dispute* and *condemn* have rather similar subjects: ‘*work on nuclear weapon’*. Generalizing two communicative actions with their subjects follows the rule: generalize communicative actions themselves, and ‘attach’ the result to generalization of their subjects as regular sub-tree generalization. Two communicative actions can always be generalized, which is not the case for their subjects: if their generalization result is empty, the generalization result of communicative actions with these subjects is empty too. The generalization result here for the case 1 above is:

condemn^dispute [work-Iran-on-nuclear-weapon].

Generalizing two different communicative actions (Fig. 7) is based on their attributes and is presented elsewhere (Galitsky et al 2009).

T1		T2
condemn [second uranium enrichment site]	↔	*proceed [develop an enrichment site in secret]*
↓ communicative action arcs		↓
suggest [Iran is secretly working on nuclear weapon]	↔	*condemn [the work of Iran on nuclear weapon]*

which results in

condemn^proceed [*enrichment site*] <leads to> *suggest^condemn* [*work Iran nuclear weapon*]

Notice that generalization

condemn [second uranium enrichment site]	↔	*condemn [the work of Iran on nuclear weapon]*
↓ communicative action arcs		↓
suggest [Iran is secretly working on nuclear weapon]	↔	*proceed [develop an enrichment site in secret]*

gives zero result because the arguments of *condemn* from T1 and T2 are not very similar. Hence we generalize the subjects of communicative actions first before we generalize communicat∅∅ive actions themselves.

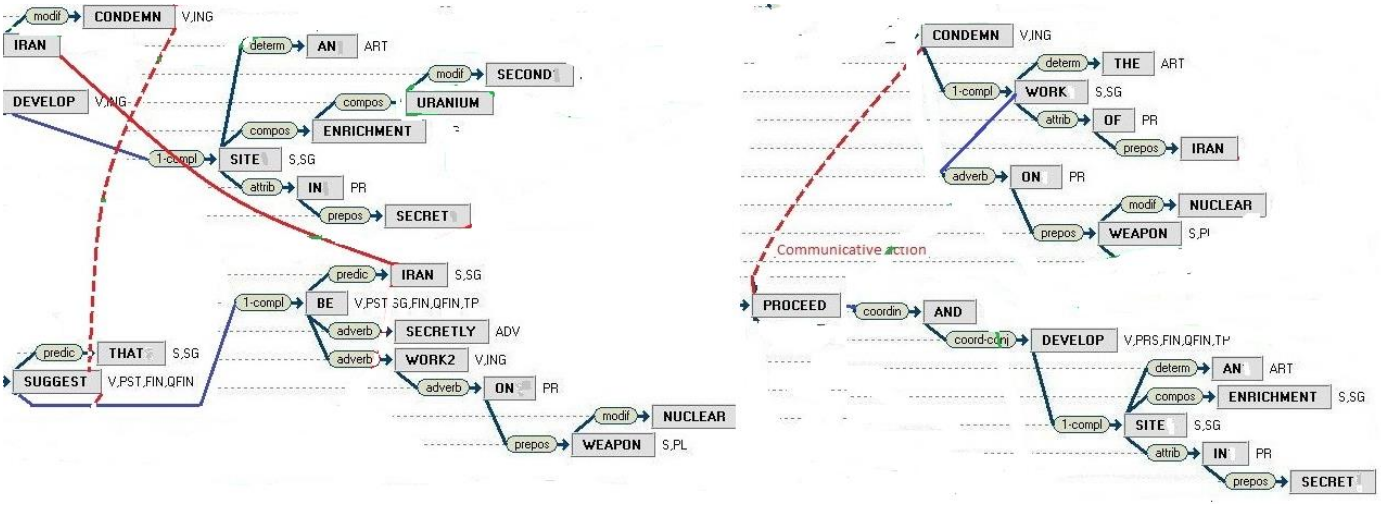

Fig.7: A fragment of PT showing the mapping for the pairs of communicative actions

We conclude the section by definition of generalization for a pair of CAs, and also for a pair of pairs of CAs.

Definition 4.2.1: $CA_{type1}(phrase_1) \wedge CA_{type2}(phrase_2) = (CA_{type1} \wedge CA_{type2})(phrase_1 \wedge phrase_2)$

when $phrase_1 \wedge phrase_2 \neq \varnothing$.

Definition 4.2.2: $(CA1_{type11}(phrase_{11}) \rightarrow CA_{type12}(phrase_{12})) \wedge (CA_{type21}(phrase_{21}) \rightarrow CA_{type22}(phrase_{22})) = CA1_{type11}(phrase_{11}) \wedge (CA_{type21}(phrase_{21}) \cup CA_{type12}(phrase_{12})) \wedge (CA_{type22}(phrase_{22}))$.

5. Computing maximal common sub-PTs

We have shown how to compute similarity between PTs via phrases. That was an approximation of finding similarity between two graphs, considering their selected paths (which correspond to phrases). Now we focus on information loss-free approach, where we compute similarity between two graphs in a classical, (linguistic) domain-independent way. Similarity between graphs is measured by the size of the maximal common subgraph (Koch 2001).

To find maximal subPT we use a reduction of the maximal common subgraph problem to the maximal clique problem (Moon and Moser 1965, Bron and Kerbosch, 1973) by constructing the edge product. The main difference with the traditional edge product is that instead of requiring the same label for two edges to be combined, we require non-empty generalization results for these edges. Though computationally not optimal, this approach is convenient for prototyping. Although for the trees this problem is *O(n)*, for the general case of graphs finding maximum common sub-graphs is NP hard (Kann, 1992).

We construct an edge product PT. Let $G_1 = (V_1, E_1, \alpha_1, L_1)$ and $G_2 = (V_2, E_2, \alpha_{21}, L_2)$ be PTs with nodes *V* and edges *E*, where $\alpha : V \rightarrow L$ is a function assigning labels to the vertices, and *L* is a finite non-empty set of labels for the edges and vertices. The ***edge product*** PT $H_e = G_1 \circ G_2$ includes the vertex set $V_H = E_1 \circ E_2$ in which all edge pairs (e_i, e_j) with $1 \le i \le |E_1|$ and $1 \le j \le |E_2|$ have to produce non-empty generalization in their edge labels. Also, these edge pairs must produce non-empty generalizations in their corresponding end vertex labels. Let $e_i = (u_1, v_1, l_1)$ and $e_j = (u_2, v_2, l_2)$. The labels coincide if $l_1 \cap l_2 \neq \varnothing$ and $\alpha_1(v_1) \cap \alpha_2(v_2) \neq \varnothing$ and $\alpha_1(u_1) \cap \alpha_2(u_2) \neq \varnothing$.

There is an edge between vertices $e_H, f_H \in V_H$ where $e_H = (e_1, e_2)$ and $f_H = (f_1, f_2)$ if two edge pairs are compatible, meaning that they are distinct: $e_1 \neq f_1$ and $e_2 \neq f_2$. Also, either condition for these edges of resultant PT should hold:

- e_1, f_1 in G_1 are connected via a vertex of the label which produces non-empty generalization with the label of the vertex shared by e_2, f_2 in G_2: $\{vertices\ for\ e_1, f_1\} \cap \{vertices\ for\ e_2, f_2\} \neq \varnothing$. We label and call them *c-edges*, or
- e_1, f_1 and e_2, f_2 are not adjacent in G_1 and in G_2, respectively. We label and call them ***d-edges***.

To get a common subPT in G_1 and G_2 each edge pair in G_1 and G_2 (vertex pair in H_e) has to have generalizable label with all other edge pairs in G_1 and G_2 (vertex pair in H_e), which are forming a common subgraph. In this way a clique in H_e corresponds to common subPTs G_1 and G_2.

A subset of vertices of *G* is said to be a clique if all its nodes are mutually adjacent. A maximal clique is one which is not contained in any larger clique, while a maximum clique is a clique having largest cardinality (Vismara and Valery 2008). After finding all **maximal** cliques for all pairs (representing question and answer, for instance) we look through all results and range them due to cliques cardinality. In this case the more edge pairs the result of generalization of two paragraphs contains the higher the relevance between two portions of text is.

Applying some effective (taking into account specific features of thicket graphs) approaches to common subgraph problem is a subject of our future work. Also we are going to use more complex types of generalization such as pattern structures (Ganter and Kuznetsov, 2001) instead of pair-wise generalization.

The clique problem is the computational problem of finding a maximum clique, or all cliques, in a given graph, which

is NP-complete, one of Karp's 21 NP-complete problems (Karp 1972). We build the edge product algorithm for PT case ouselves and integrated it with the available solution for the click problem of JGraphT library.

After finding all maximal common subgraph for all pairs (representing question and answer for instance) we rank results according to the score, based on size of common substructure and its linguistic properties. Each part of speech has its own score calculated from seacrch engine statistics.

6. Architecture of PT processing system

Application of PT generalization for search occurs according to the following scenario. For the question and candidate answer, we build a pair of PTs. Then we perform generalization of PTs, either without loss of information, finding a maximum common PT subgraph, or with the loss of information, approximating the paths of resultant subgraph by generalizing thicket phrases. Search relevance score is computed accordingly as a total number of vertexes in a common maximum subgraph in the first case, and calculating the number of words in maximal common sub-phrases, taking into account weight for parts of speech (Galitsky et al 2012), in the second case. This scenario will be evaluated in details in the section to follow.

The system architecture is depicted in Fig. 8. There are three system components, which include Parse Thicket building, phrase-level processing, and graph processing.

The textual input is subject to a conventional text processing flow such as sentence splitting, tokenizing, stemming, part-of-speech assignment, building of parse trees and coreferences assignment for each sentence. This flow is implemented by either OpenNLP or Stanford NLP, and the parse thicket is built based on the algorithm presented in this paper. The coreferences and RST component strongly relies on Stanford NLP's rule-based approach to finding correlated mentions, based on the multi-pass sieves.

The graph-based approach to generalization relies on finding maximal cliques for an edge product of the graphs for PTs being generalized. As it was noticed earlier the main difference with the traditional edge product is that instead of requiring the same label for two edges to be combined, we require non-empty generalization results for these edges. Hence although the parse trees are relatively simple graphs, parse thicket graphs reach the limit of real-times processing by graph algorithms.

The system architecture serves as a basis of OpenNLP – similarity component, which is a separate Apache Software foundation project, accepting input from either OpenNLP or Stanford NLP. It converts parse thicket into JGraphT (http://jgrapht.org/) objects which can be further processed by an extensive set of graph algorithms. In particular, finding maximal cliques is based on (Bron and Kerbosch, 1973) algorithm. Code and libraries described here are also available at http://svn.apache.org/repos/asf/opennlp/sandbox/opennlp-similarity/. The system is ready to be plugged into Lucene library to improve search relevance. Also, a SOLR request handler is provided so that search engineers can switch to a PT-based multi-sentence search to quickly verify if relevance is improved.

The second and third components are two different ways to assess similarity between two parse thickets. Phrase-level processing for the phrases of individual sentences has been described in detail in our previous studies (Galitsky *et al.*, 2013a). In this study we collect all phrases for all sentences of one paragraph of text, augment them with thicket phrases (linguistic phrases which are merged based on the inter-sentence relation), and generalize against that of the other paragraph of text.

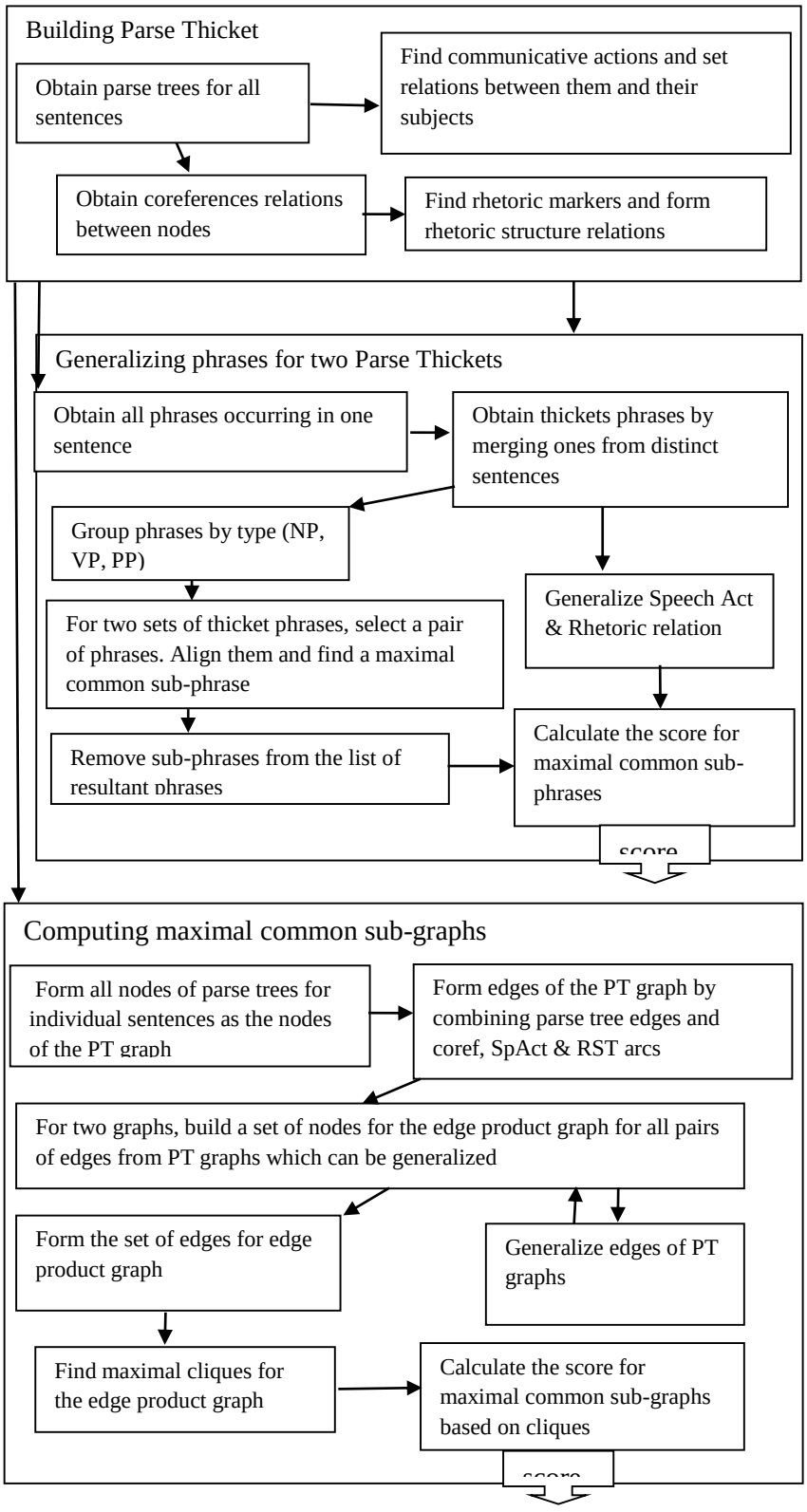

Fig. 8: Architecture of Parse Thicket processing system.

7. Evaluation of PT-supported search relevance

PTs and their generalizations are important for domain-independent text relevance assessment. We propose a number of evaluation scenarios to compare the relevance performance of PT-supported search and its various forms of reduction. We demonstrate that simplifying construction of PT, discarding one or

another linguistic feature, we lose relevance. We also compare the performance of complete PT-supported search with other state-of-the-art question answering systems dealing with multi-sentence queries and /or multi-sentence questions.

In our earlier studies we explored the following cases for how generalization supports question answering:

- Single sentence question against single sentence answer (Galitsky et al 2012).
- Single sentence question against multiple sentences in the answer by pair-wise generalization and summing up the score (Galitsky et al 2013).

We will re-evaluate these cases in the unified framework to compare the results with the focus of the current paper, paragraph-sized question against a paragraph-sized answer (or its snippet).

In this section we demonstrate that pair-wise sentence generalization approach lacking discourse features can be outperformed by PT approach.

Having formed a pair of PTs for question and answer, we will compare:

- Pair-wise sentence-sentence generalization (ignoring inter-sentence links) versus full PT generalization
- Phrase-based approximation of generalization versus finding maximum common sub-parse thickets.
- Contribution of two sources of discourse structure: RST and SpActT for search relevance
- Role of PT-supported search for various domain, from product search to social.

In terms of complexity of queries and answers, we will proceed from single sentences queries/answers to multiple sentences queries/answers..

7.1 Evaluation settings

We conducted evaluation of relevance of syntactic generalization – enabled search engine, based on Bing search engine APIs. Instead of maintaining search index ourselves, for the purpose of evaluation we relied on Bing index and baseline search relevance. In terms of reproducibility of the experimental results in this study, since we measure a relative relevance improvement compared to Bing's baseline, once we have a fixed publically available set of queries, it is acceptable. From Bing search results, we get page titles, snippets, and also extract paragraphs of text from the original webpage.

For an individual query, the relevance was estimated as a percentage of correct hits among the first thirty, 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 30 search results obtained for a query by Bing API, and then re-ranked them according to the score of the given search setting (syntactic generalization score).

The list of products which serves the basis of queries is available at https://code.google.com/p/relevance-based-on-parse-trees/downloads/detail?name=Queries900set.xls. We took each product and found a posting somewhere on the web (typically, a blog or forum posting) about this product, requesting a particular information or addressing a particular user feature, need, or concern. From such extended expression containing product names, we formed the list queries of desired complexity.

To estimate the statistical significance of results of relevance improvement, we estimate the standard deviation σ_Δ of Δ, the difference between the baseline average relevance and the one obtained by a particular re-ranking. For the evaluation set of 40 search sessions for both baseline and AFP-supported search, we have

$$\sigma_\Delta = \sqrt{\sigma_{baseline}^2/40 + \sigma_{PT}^2/40}.$$

To do that, we assume that the search accuracy can be described by a normal distribution, and the number of searches is large enough (Kohavi 1995).

7.2 Pair-wise sentence generalization for question-answer similarity

There are a number of limitations in how the 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, related to the same entity. Our first evaluation setting, the pair-wise matching of parse tree for questions and answers, addressing this issue and showing that once parse trees are taken into account in addition to keywords, relevance is increasing. In our previous studies we already demonstrated this fact, and in this project we repeat this evaluation setting to conform to the consecutive ones based on PTs.

Table 5 shows the search relevance evaluation results for single-sentence answers. The first and second columns show the types of phrases/sentences serving as queries. The third column shows the baseline Bing search relevancy. The fourth and fifth columns shows relevance of re-ranked search for snippets and original paragraph(s) from the webpage, and the sixth column shows relevance improvement compared with the baseline.

One can observe that the higher the query complexity, the higher the impact of generalization for question and answer for re-ranking. For the simplest queries, there is no improvement. For 5-10 keywords phrases improvement starts being noticeable, of 4% and reaches 7% and 8% for two/three sentence queries respectively. As the absolute precision of search naturally drops when queries become more complex, relative contribution of syntactic generalization increases.

In most cases, using original text is slightly better than using snippets, by about 0.5% except the case of the simple queries. In the case of 3-5 keywords the use of generalization is unreasonable, so the results can be even distorted and re-ranking by generalization score is not meaningful.

Table 5: Evaluation of pairwise-sentence generalization search

Query	Answer	Relevancy of baseline Bing search, %, averaging over 40 searches	Relevancy of re-sorting by pair-wise sentence generalization with **snippets**, %, averaging over 40 searches	Relevancy of re-sorting by pair-wise sentence generalization with **text on original page**, %, averaging	Relevancy improvement: re-sorted relevance /for Bing	Standard Deviation For the relevancy improvement
3-4 word phrases	1 sentence	87.9	88.3	90.3	1.016	0.0054
	2 sentences	83.7	81.8	84.2	0.992	0.0048
	3 sentences	79.9	79.5	80.7	1.003	0.0056
	Average	83.83	83.20	85.07	1.00	
5-10 word phrases to a sentence	1 sentence	82.9	84.5	84.4	1.019	0.0061
	2 sentences	79.5	83.1	83.7	1.049	0.006
	3 sentences	77.5	82.2	83.2	1.067	0.0065
	Average	79.97	83.27	83.77	1.04	
2 sentences, and in each:	1 sentence	66.3	69.5	70.8	1.058	0.0052
	2 sentences	65.2	71	70.5	1.085	0.0061
	3 sentences	65.4	70.2	70.9	1.079	0.0058
	Average	65.63	70.23	70.73	1.07	
3 sentences, and in each:	1 sentence	62.1	67.3	68.1	1.090	0.0046
	2 sentences	60.4	65	65.4	1.079	0.0054
	3 sentences	59.9	64.7	64.2	1.076	0.0048
	Average	60.80	65.67	65.90	1.08	

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.

7.3 Single sentence query and answer distributed through multiple sentences

It is hard for modern search engines to determine for a given occurrence of keywords, being in different sentences, if they are related to each other, and are related to the query term. Once we establish links

between words in different sentences, it should help in determining how these words are potentially connected with the query keywords. We will evaluate this observation in this section.

Relevance improvement for the queries ranging from a few keywords to a whole sentence is shown in Table 6. One can see that the simplest cases of short query and a single compound sentence gives about 3% improvement. PT-based relevance improvement stays within 6% as query complexity increases by a few keywords, and then increases to 12% as query becomes a whole sentence. For the same query complexity, naturally, search accuracy decreases when more sentences are required for answering this query. However, contribution of PTs does not vary significantly with the number of sentences the answer occurs in (one, two , or three). PT-supported search outperforms the pairwise generalization-supported search for the similar setting of complex phrase to one sentence query (Table 5 rows 6-10) by 3- 8%.

We separately measure search relevance when PT is same/sub/super-entity - based, coreferences-based, RST-based and SpActT-based (columns 4-7). Our hybrid approach includes all these sources for links altogether (column 8). 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 PT improved the search relevance, compared to original search results of Bing.

We found that contribution of inter-sentence links decreases in the following order: *coreferences, same/sub/super-entity, RST,* and *SpActT* for simpler queries, *and same/sub/super-entity, SpActT, coreferences, RST* for the full sentence queries; the deviation between their contribution is within a percent or two. What is visible is that all of these sources of discourse information in a stand-alone mode improve the relevance, hence their hybrid is indeed required for overall PT-supported search. This conclusion is made taking into account that these sources are independent relevance improvers, since they are based on mechanisms and linguistic theories of totally different natures.

Table 6: Search improvement results for PT approach: single sentence query and multi-sentence answer

Query	Answer	Relevancy of baseline Bing search, %, averaging over 40 searches	Relevancy of re-sorting by PT generalization based on **coreferences**, %,	Relevancy of re-sorting by PT generalization based on **same/sub/super-entity**, %,	Relevancy of re-sorting by PT generalization based on **RST**, %,	Relevancy of re-sorting by PT generalization based on **SpActT**, %,	Relevancy of re-sorting by hybrid **coreference+entity+RST+Sp**	Relevancy **improvement** for PT approach, comp. to baseline	Standard Deviation for relevancy improvement
3-4 word phrases	1 sentence	81.5	82.8	82.7	82.1	81.5	83.4	1.023	0.0091
	2 sentences	79	79.3	78.5	79.7	80.2	81.7	1.034	0.0087
	3 sentences	76.2	78.4	78.9	75.2	77.2	78.9	1.035	0.0090
	Average	78.90	80.17	80.03	79.00	79.63	81.33	1.031	
5-10 word phrases	1 sentence	78.5	79.5	80.4	77.5	78.9	81.2	1.034	0.0092
	2 sentences	75.7	78.7	78.2	78	75.1	80	1.057	0.0095
	3 sentences	72.2	77.9	76.8	74.2	73.3	80.6	1.116	0.0082
	Average	75.47	78.70	78.47	76.57	75.77	80.60	1.069	
1 sentence	1 sentence	76.2	78.7	81.3	78.9	79.1	82.1	1.077	0.0079
	2 sentences	72.9	75.5	76.3	74.5	77	81.7	1.121	0.0087
	3 sentences	70.3	73.9	75.1	71.2	75.4	81.4	1.158	0.0081
	Average	73.13	76.03	77.57	74.87	77.17	81.73	1.12	

We proceed to evaluation of how generalization of PTs can improve multi-sentence search, where one needs to compare a query as a paragraph of text against a candidate answer as a paragraph of text (snippet). Evaluation results show the accuracies in percentages, averaging over 100 searches.

7.4 Query is a paragraph and answer is a paragraph

When the query complexity increases from one sentence to four sentences, the contribution of PT increases from 12% for a single sentence to 13% for double sentence and then to 14% for triple sentence and stays the same for four sentences in the query. Although as queries become more complicated, overall drop of PT-supported relevance is not lower than the respective drop of baseline relevance, the significance of the relevance improvement is obvious (Table 7).

We observe that contribution of inter-sentence links decreases in the following order: *SpActT* , *coreferences, same/sub/super-entity, and RST,* and for two sentence queries, and *SpActT, RST, coreferences,* and *same/sub/super-entity*. Hence for longer queries and answers, the role of discourse theories is higher than that of for simpler queries, where *coreferences*, and *same/sub/super-entity* turned out to be more important.

RST for the full sentence queries; the deviation between their contribution is within a percent or two.

Table 7: Relevancy improvement for query and answers as paragraphs

Query	Answer	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by PT generalization based on **coreferences**, %,	Relevancy of re-sorting by PT generalization based on **same/sub/super-entity**, %,	Relevancy of re-sorting by PT generalization based on **RST**, %,	Relevancy of re-sorting by PT generalization based on **SpActT**, %,	Relevancy of re-sorting by hybrid **coreference+entity+RST+Sp**	Relevancy **improvement** for PT approach, comp. to baseline	Standard Deviation for relevancy improvement
2 sentences	2 sentences	75.1	76.8	76.7	75.1	81	83.4	1.111	0.0089
	3 sentences	71.1	75.3	72.5	73.7	78.7	82.7	1.163	0.0091
	4 sentences	72.1	75.4	74.9	74.2	75.2	80.9	1.122	0.0079
	Average	72.77	75.83	74.70	74.33	78.30	82.33	1.132	
3 sentences	2 sentences	72.1	75.5	73.4	74	74.9	81.5	1.130	0.0087
	3 sentences	70.9	74.7	72.2	73.1	75.1	79.9	1.127	0.0078
	4 sentences	67.1	72.9	70.8	71.2	73.3	79.3	1.182	0.0083
	Average	70.03	74.37	72.13	72.77	74.43	80.23	1.146	
4 sentences	2 sentences	67.7	70.7	69.3	71.9	72.1	76.1	1.124	0.0085
	3 sentences	65.4	70.5	67.2	71.5	73.7	74.7	1.142	0.0092
	4 sentences	62.1	68.9	64	69.2	71	72.4	1.166	0.0087
	Average	65.07	70.03	66.83	70.87	72.27	74.40	1.144	0.0083

7.5 Phrase-based and graph-based implementation of generalization

Once we know that we should leverage all available discourse-related information for better search relevance, we investigate various approaches to generalization computation, and also explore performance in various domains where search is associated with multi-sentence questions and multi-sentence answers. Moreover, for each domain and query/answer complexity, once we obtain search results from Bing Search API, we either take its snippet or download the original webpage and extract the respective paragraph, to form its PT.

Three domains are used in evaluation in this section (Table 8):

- Product recommendation, where a user explicitly or implicitly (according to a belief of an automated agent) is requesting information about products. This agent finds a chat, and determines user intent from text (analyzing epistemic states of this user, and mention) that this user is seeking a product-related recommendation. The input for such the recommendation is a few sentences of text. Given this input, a search request is sent to Bing Blogs and Forums and candidate recommendations are filtered out in terms of relevance by PT generalization. The data is collected from eBay.com and StubHub.com
- Travel recommendation, where an agent reads chats about travel activities, things to do and hotels. The agent tracks travel chats, forums, blogs and trip reports and produces recommendation when requested or when user intent is detected.The data was collected at UpTake.com acquired by Groupon.com.

- Facebook automated posting agent, which issues posts on behalf of its human host. For the purpose of promoting social activity and enchance communications with the friends other than most close ones, this agent is authorized to comment on postings, images, videos, and other media. Given one or more sentence of user posting or image caption, this agent issues a Bing Web and Bing Blogs APIs search request and filters the results for relevance. Experiments with Facebook account were conducted using OpenGraph involving a number of personal accounts (Fig. 9).

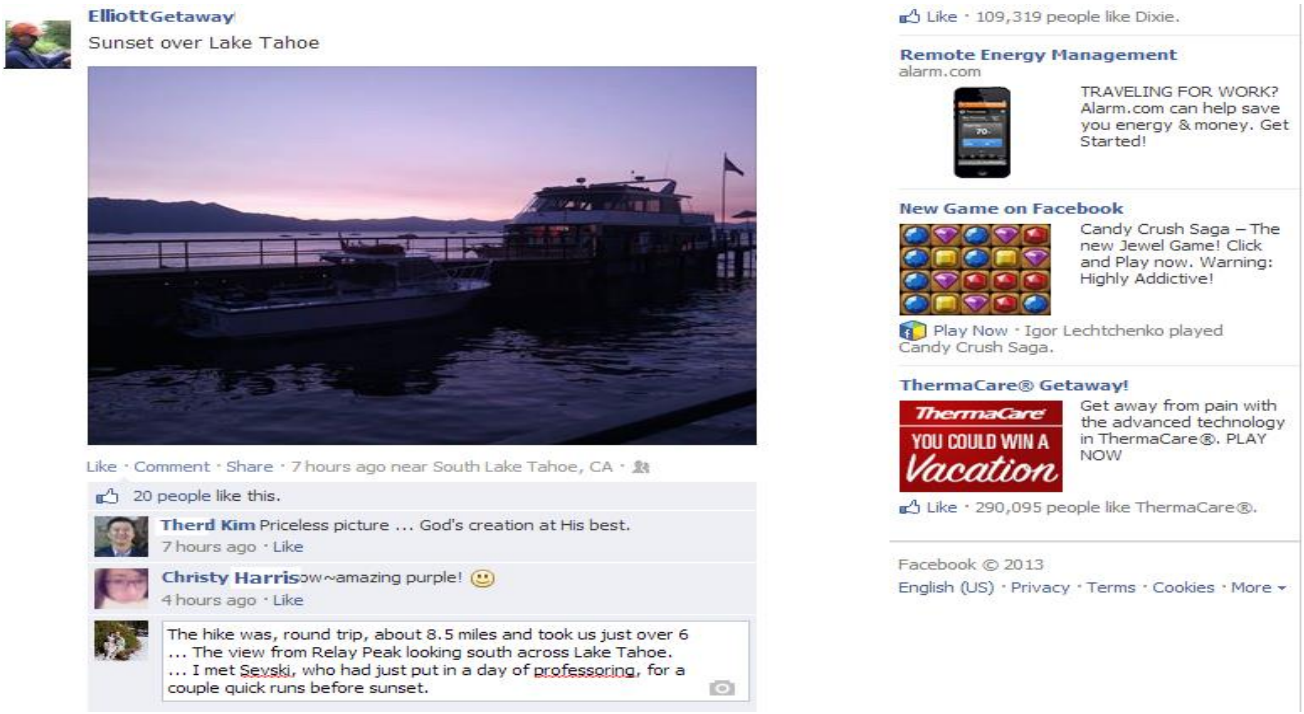

Fig. 9: The Facebook agent is posting a message on behalf of its human host. This message is obtained from Bing Blogs API and is supposed to be relevance (unlike the most of Facebook ads on the right).

The value of PT based generalization varies from domain to domain significantly, mostly depending on the writing style and use of terminology by the authors presenting their opinions on the products. When things in a domain are named uniquely, and the typical writing style is plain enumeration of product features, contribution of PT is the least (shopping product domains). On the contrary, where writing styles vary a lot and different people name the same things differently, in such horizontal domain as Facebook, the baseline relevance is low, the resultant relevance is lower (63%) than in the other domains (73-75).

One can see that contribution of SpAct source varies from domain to domain. In product and travel recommendations, its contribution is low (about 1%). At the same time, in the Facebook domain, where the description of interaction between people do occur, its contribution reaches 3%.

Proceeding from Snippets to original paragraph(s) in a webpage gives further 0.8% increase for both thicket phrase-based and graph-based computation of PT.

Graph-based algorithms (Bunke et al 2002, Conte et al 2007) for PT generalization are associated with much higher complexity than phrase based. At the same time, loss of information for a phrase base approach leads to less than 1% drop in the resultant relevance. Hence we select the phrase-based approach to PT generalization as most efficient.

Table 8: Evaluation results for various search domains and for various implementations of PT generalization

Query type	Query complexity	Relevance of baseline Bing search, %, averaging over 40 searches	Relevance of PT/**phrase** generalization search, %, averaging over 40 searches, using **original text, without SpAtcT**	Relevance of PT/**phrase** generalization search, %, averaging over 40 searches, **using snippets**	Relevance of PT/**phrase** generalization search, %, averaging over 40 searches, using **original text**	Relevance of PT/ **graph** generalization search, %, averaging over 40 searches, using **snippets**	Relevance of PT/ **graph** generalization search, %, averaging over 40 searches, using **original text**
Shopping/ product recommend ation	1compound sentence	63.3	71.9	73.4	73.1	74.1	73.3
	2 sent	61.5	70.8	71.9	71.7	72.3	73.6
	3 sent	59.9	70.9	72	72.5	73.7	74.4
	4 sent	60.4	66.7	68.5	69.6	70.1	70.7
	Average	61.275	70.075	71.45	71.725	72.55	73
Travel recommend ation	1compound sent	64.8	69.2	71.6	73.1	74.8	75.2
	2 sent	60.6	71.7	73.1	73.9	74.6	75.5
	3 sent	62.3	68.1	70.9	72.6	72	72.9
	4 sent	58.7	72.7	71.5	72.6	73.4	74.7
	Average	61.6	70.425	71.775	73.05	73.7	74.575
Facebook friend agent support	1compound sent	54.5	61.3	63.3	65.3	66.2	67.2
	2 sent	52.3	60.9	60.7	62.1	63.4	63.9
	3 sent	49.7	55.4	61.7	61.9	60.8	61.9
	4 sent	50.9	55.5	60.5	61.1	61.5	62.7
	Average	51.85	58.275	61.55	62.6	62.975	63.925

7.6 Comparison of search performance with other studies

In this section we evaluate the search task which is not specific to the given study, but constitutes a standard test-bed for question answering systems. We obtained a list of entity-based queries, some of which require an answer contained in multiple sentences. The evaluation set of questions was obtained from

http://cogcomp.cs.illinois.edu/Data/QA/Trec10questions.txt (Li & Roth 2002) and

http://cogcomp.cs.illinois.edu/Data/QA/QC/train_1000.label.

We took queries from this list and converted into short phrases, longer phrases, and extended by 1-2 sentences, to match the above evaluation cases. There are also 40 search sessions per query/ answer types.

These search evaluation settings for Table 6 were inspired by TREC 2010, whose goal was to perform entity-oriented search tasks on the web. Many user information needs concern entities (people, organizations, locations, products, etc.) which are better answered by returning specific objects instead of just any type of documents. Like Trec 2010 Entity track, we used normalized Discounted Cumulative Gain (NDCG), the normalized discounted cumulative gain at rank R (the number of primary and relevant search results for that topic) where primary pages get gain 3 and relevant pages get gain 1. This is unlike our previous evaluation settings.

The intuition behind the Discounted Cumulative Gain approach is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. The discounted CG accumulated at a particular rank position p is defined as:

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$$

Where rel_1 is the relevance of the first result, and rel_i is the relevance of the consecutive results $2<i<10$. Dividing the obtained DCG by the DCG of the ideal ranking, we obtain a normalized DCG.

TREC evaluation was conducted in a similar environment, relying on Bing, Yahoo, and other web knowledge sources.

In the example search result Fig. 10. PT-supported search re-ranks the snippets to find the entity occurring in an expression [ENTITY is *<query with substituted WH-word>*]. In most cases, ENTITY occurs in a different sentence to the one which is best matched with *<query with substituted WH-word>*.

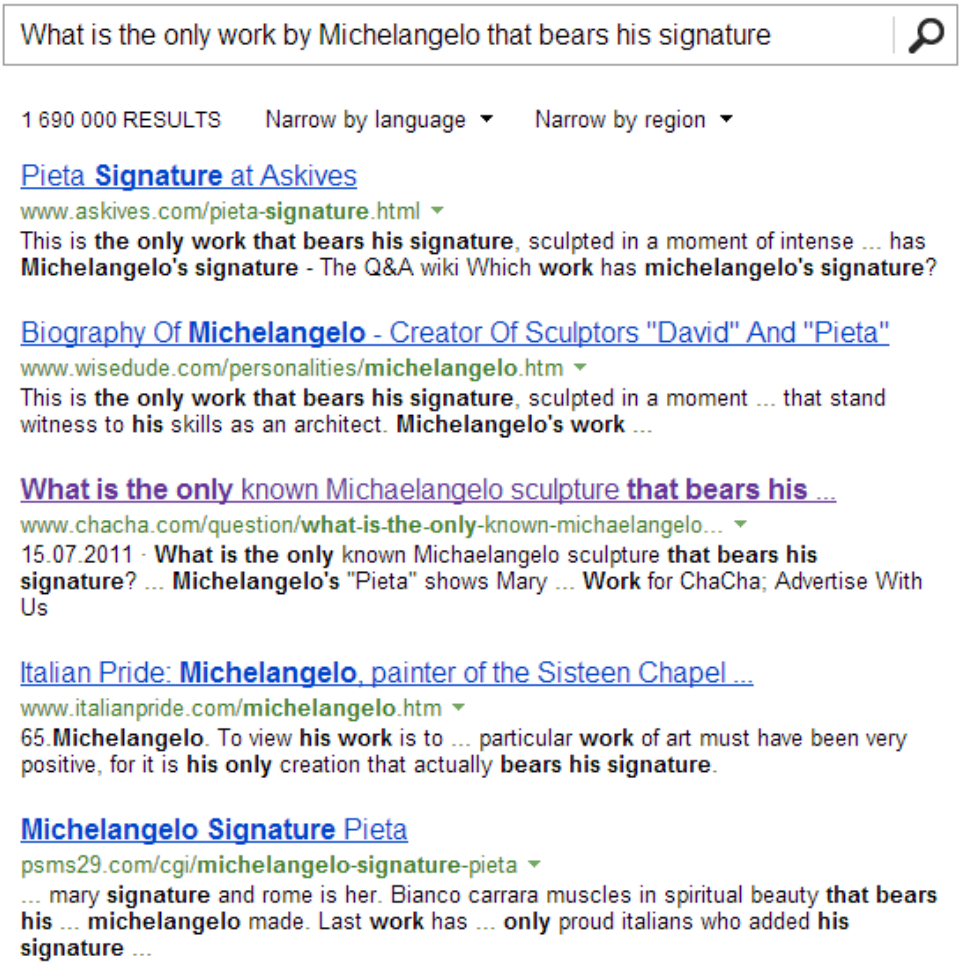

Fig. 10: Bing search results for the entity-based query with expected answer “Michelangelo’s Pieta“, obtained by APF-supported search

Table 9: Entity-based search evaluation

Query	Answer	Relevancy of baseline Yahoo search, NDCG@R, averaging over 20 searches	Relevancy of baseline Bing search, NDCG@R, averaging over 20 searches	Relevancy of re-sorting by pair-wise sentence generalization, NDCG@R	Relevancy of re-sorting by hybrid RST+SpActT forest generalization, NDCG@R	Relevancy improvement for parse forest approach, comp. to pair-wise generalization	Standard Deviation for relevance improvement	$\sigma_\Delta = \sqrt{\sigma_{baseline}^2/40 + \sigma_{APP}^2/40}$.
3-4 word phrases	1 sentence	0.2771	0.2830	0.3006	0.3185	1.1373	0.0298	0.00178
	2 sentences	0.2689	0.2759	0.3183	0.3065	1.1251	0.0282	0.00139
	3 sentences	0.2649	0.2600	0.2680	0.3058	1.1651	0.0324	0.00182
	Average	0.2703	0.2730	0.2957	0.3102	1.1422	0.0287	
5-10 word phrases	1 sentence	0.2703	0.2612	0.2910	0.3062	1.1523	0.0289	0.00214
	2 sentences	0.2641	0.2583	0.2761	0.3117	1.1933	0.0321	0.00183
	3 sentences	0.2566	0.2602	0.2620	0.2908	1.1253	0.0297	0.00219
	Average	0.2636	0.2599	0.2764	0.3029	1.1570	0.0315	
1 sentence	1 sentences	0.2724	0.2674	0.2790	0.3123	1.1570	0.0280	0.00131
	2 sentences	0.2535	0.2580	0.2742	0.3061	1.1970	0.0285	0.00192
	3 sentences	0.2469	0.2444	0.2606	0.3057	1.2444	0.0284	0.00254
	Average	0.2576	0.2566	0.2713	0.3080	1.1981	0.0307	
2 sentences	1 sentence	0.2593	0.2557	0.2776	0.2888	1.1217	0.0294	0.00141
	2 sentences	0.2516	0.2421	0.2615	0.2766	1.1207	0.0312	0.00129
	3 sentences	0.2331	0.2530	0.2633	0.2885	1.1872	0.0307	0.00165
	Average	0.2480	0.2502	0.2675	0.2846	1.1427	0.0311	

We compare the results of PT-based search with that of the TREC Entity Track participants (Table 9). The best teams, BIT (Jiang et al 2010) and FDWIM2010, had slightly higher relevance, NDCG 0.3694 and 0.2726, respectively, and the rest of the teams, including Purdue, NiCT , ICTNET, and UWaterlooEng obtained a lower relevance compared to APF-based approach (Balog et al 2010). In the current study for the hybrid RST+SpActT forest generalization approach we obtained NDCG 0.3123, 0.3061, 0.3057 for 1-sentence answers, 2-sentence answers and 3-sentence answers respectively. It worth mentioning that the above approaches are oriented at answering entity-based questions whereas the current approach targets the cases where found keywords are distributed through multiple sentenced in the

search result snippet. This evaluation covers the overlap of these two cases, and we believe PT performance is satisfactory here.

For the entity based search from the TREC Entity Track queries the improvement of search by using PT and especially RST is higher than for the search in Section 4, for both short and long queries. This is due to the fact that entity-based questions take advantage of the coreferences and RST relations such as elaboration, which are typical in the paragraphs of text answering entity-based questions. Since both short and long entity-based phrases heavily rely on the coreferences and RST, there is a relatively uniform improvement of search accuracy, compared to the search in previous evaluation subsections where PT contribution for more complicated questions is higher.

Over the past few years, complex questions have been the focus of much attention in the automatic question-answering community. Most current complex QA evaluations included the 2004 AQUAINT Relationship QA Pilot, the 2005 TREC Relationship QA Task, and the TREC 2010 entity task ((Jiang et al 2010) whose results we compared to ours), 2006 and 2007 Document Understanding Conference (DUC). These evaluations require systems to return unstructured lists of candidate paragraph-length answers in response to a complex question that are responsive, relevant, and coherent. For the DUC settings, (Chali et al 2009) reports 2% improvement (from 0.44984 to 0.45762) of parse tree similarity-based approach over keyword-based (lexical) for the K-means framework (Cahel et al 2009) which roughly corresponds to our improvement of single sentence generalization-based search over the baseline (Section 7.2).

(Moschitti and Quarteroni 2011) report the accuracy (F1 measure) on the TREC-QA dataset of 24.2 ± 3.1% for a bag-of-words classifier and 39.1 ± 6.9% for the optimal combination of tree kernels. If one re-ranks (deteriorates) search engine search results by keyword occurrence only, and then compare with the APF performance, a similar performance would be observed. Single-sentence generalization relies on similar linguistic information about questions and answering as tree kernel, and APF extension becomes noticeable compared to commercial search engine results rather than bag-of-words systems.

8. Related work

General pattern structures consist of objects with descriptions (called patterns) that allow a semi-lattice operation on them (Ganter & Kuznetsov 2001). In our case, for paragraphs of text to serve such objects, they need to be represented by structures like parse thickets, which capture both syntactic level and discourse-level information about texts. Pattern structures arise naturally from ordered data, e.g., from labeled graphs ordered by graph morphisms. In our case labeled graphs are parse thickets, and morphisms are the mappings between their maximal common sub-graphs. Besides finding maximal common sub-graph, PTs can be viewed from the standpoint of graph search problem. For example, (Mandow 2001) describes a new general algorithm for graph search problems with additive lexicographic goals. The use of lexicographic goals in the formulation of search problems provides greater control and expressive power over the properties of solution paths in the algorithm, called METAL-AN*.

One of the first systems for the generation of conceptual graph representation of text is described in (Sowa and Way, 1986). It uses a lexicon of canonical graphs that represent valid (possible) relations between concepts. These canonical graphs are then combined to build a conceptual graph representation of a sentence. Since then syntactic processing has dramatically improved, delivering reliable and efficient results. PTs can be viewed as conceptual graphs which are derived automatically from Concept mining has found a number of applications as well (Bichindaritz, & Akkineni 2006).

Hensman & Dunnion (2009) describes a system for constructing conceptual graph representation of text by using a combination of existing linguistic resources (VerbNet and WordNet). However, for practical applications these resources are rather limited, whereas syntactic level information such as syntactic parse

trees is readily available. Moreover, building conceptual structure from individual sentences is not as reliable as building these structures from generalizations of two and more sentences.

In this study we attempt to approach conceptual graph level (Sowa 1984, Polovina & Heaton 1992) using pure syntactic information such as syntactic parse trees and applying learning to it to increase reliability and consistency of resultant semantic representation. The purpose of such automated procedure is to tackle information extraction and knowledge integration problems usually requiring deep natural language understanding (Galitsky 2003) and cannot be solved at syntactic level.

Whereas machine learning of syntactic parse trees for individual sentences is an established area of research, the contribution of this paper is a structural approach to learning of syntactic information at the level of paragraphs. A number of studies applied machine learning to syntactic parse trees (Collins & Duffy 2002), convolution kernels being the most popular approach (Haussler 1999). Parse tree kernels are also used for plagiarism detection (

(Harabagiu et al. 2006) introduce a new paradigm for processing complex questions that relies on a combination of question decompositions (based on a Markov chain), factoid QA techniques, and multi-document summarization. The Markov chain is implemented by following a random walk with a mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and sub-questions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art QA system in order to retrieve a set of passages that can later be merged into a comprehensive answer. The authors show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions. This approach does not rely on the association between concepts available for decomposed (simpler) questions from the commercial search engines. In the current study we achieve relevance based on better match of questions to answers, obtained by search engines which have learned the best matches for decomposed questions relying on user selections. Hence in our evaluation settings we skip decomposition and do not do summarization, achieving higher relevance by finding relevant documents among the candidate set which has been formed by search engine APIs. Moreover, to the best of our knowledge, no approach to answering complex questions is relying on linguistic discourse theories.

The evaluations in (Harabagiu et al. 2006) have shown that the question decompositions lead to more relevant and complete answers. Moreover, the coverage of auto generated question decompositions, when compared with the questions generated from the answer summary, are better indicators of answer quality than the relevance score to the complex question. The question coverage for automatic methods is 85% of the coverage of questions produced by humans. Within the framework of the current study, question decomposition occurs via matching with various parse trees in the answers. If the baseline answers are re-ranked by humans, the coverage (recall) is about 18% higher than the automated system in the lower section of Table 5 (where 3-sentence questions are matched with 3-sentence answers, recall values are not shown in the table).

Use of PT allows for domain-independent search applications, desired more than three decades ago. (Xing & Li 1992) proposes a domain-independent approach to constructing natural language interfaces for applications. Domain independence provides generality to natural language interfaces. A hierarchical structure for natural language processing and a representation language, the Intermediate Carrier Language, are introduced. A special phase of natural language processing called domain compiling is also proposed and described.

Learning of PTs can be viewed as learning cases in the framework of case-based reasoning. Richter 2009 compares case-based reasoning with other methods searching for knowledge, considering it as a resource that can be traded. It has no value in itself; the value is measured by the usefulness of applying it in some process. Such a process has info-needs that have to be satisfied. The concept to measure this is the economical term utility. In general, utility depends on the user and its context, i.e., it is subjective. The

author introduces the levels of contexts from general to individual and illustrates that Case-Based Reasoning on the lower, i.e., more personal levels is quite useful, in particular in comparison with traditional informational retrieval methods.

(Yang and Soo 2012) develop a technique to extract conceptual graphs from a patent claim using syntactic information such as POS and dependency tree, as well as semantic information such as the background ontology (Vicient et al 2013). Due to extensive technical domain terms and long sentences in patent claims, it is difficult to apply a NLP Parser directly to parse the plain texts in the patent claim. This paper combines techniques such as finite state machines, Part-Of-Speech tags, conceptual graphs, domain ontology and dependency tree to convert a patent claim into a formally defined conceptual graph. The method of a finite state machine splits a lengthy patent claim sentence into a set of shortened sub-sentences so that the NLP Parser can parse them one by one effectively. The Part-Of-Speech and dependency tree of a patent claim are used to build the conceptual graph based on the pre-established domain ontology. The result shows that 99% sub-sentences split from 1700 patent claims can be efficiently parsed by the NLP Parser.

9. Conclusions

Whereas machine learning of syntactic parse trees for individual sentences is an established area of research (Haussler, 1999 ; Collins and Duffy, 2002; Moschitti, 2006), the contribution of this paper is a structural approach to learning of syntactic information at the level of paragraphs. Galitsky (2012; 2013) observed how employing a richer set of linguistic information, such as syntactic relations between words, assists relevance tasks. To take advantage of semantic discourse information, we introduced parse thicket representation and proposed the way to compute similarity between texts based on generalization of parse thickets. In this work we build the framework for generalizing PTs as sets of phrases to re-rank search results obtained via keyword search.

The operation of generalization to learn from parse trees for a pair of sentences turned out to be important for search re-ranking. Once we extended it to learning parse thickets for two paragraphs, we observed that the relevance is further increased compared to the baseline (Bing search engine API), which relies on keyword statistics in the case of multi-sentence query. Parse thicket is intended to represent the syntactic structure of text as well as a number of semantic relations for the purpose of the real time re-ranking of the search results. Parse thickets include relations between words in different sentences, such that these relations are essential to perform a broad match of queries and answers.

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. We demonstrated that search relevance can be improved if search results are subject to confirmation by parse thicket generalization, where answers occur in multiple sentences. We showed that each source contributes on its own to improve relevance, and altogether inter-sentence links are fairly important for finding relevant complex answers to complex questions.

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 in both worlds: learn complete parse tree information augmented with an adjustment of discourse theory allowing computational treatment.

To the best of our knowledge this is one of the first works in learning a semantic discourse to solve a search relevance problem, using on a sequence of parse trees. Instead of using linguistic information of individual sentences, we can now compute text similarity at the level of paragraphs.

We have contributed the PT-based functionality to OpenNLP so that search engineers can easily plug it in their search infrastructure. The algorithms for PT construction, PT generalization via phrases and via graphs are available at https://code.google.com/p/relevance-based-on-parse-trees.

9.1 Scalability of the approach

One approach to learning parse trees is based on tree kernels, where the authors propose a technique oriented specifically to parse trees, and reduce the space of all possible sub-trees. Partial tree kernel (Moschitti, 2006) allows partial rule matching by ignoring some child nodes in the original production rule. Tree Sequence Kernel or TSK (Sun *et al.*, 2011) adopts the structure of a sequence of sub-trees other than the single tree structure, strictly complying with the original production rules. Leveraging sequence kernel and tree kernel, TSK enriches sequence kernel with syntactic structure information and enriches tree kernel with disconnected sub-tree sequence structures.

The approach of phrase-level matching for parse trees and thicket phrase matching for parse thickets turns out to be much more efficient than graph based approaches, including graph kernels. Instead of considering the space of all possible sub-graphs, we consider paths in trees and graphs, which correspond to linguistic phrases, or to phrases merged according to inter-sentence relations of the discourse. We do not consider parts of trees or thickets which correspond neither to a phrase of parse tree nor to a thicket phrase of thicket. This is due to the observation that such sub-trees and sub-graphs are incomplete, redundant or noisy features to rely on, conducting learning.

To estimate the complexity of generalization of two parse thickets, let us consider an average case with five sentences in each paragraph and 15 words in each sentence. Such thickets have on average 10 phrases per sentence, 10 inter-sentence arcs, which give us up to 40 thicket phrases each. Hence for such parse thickets we have to generalize up to 50 linguistic phrases and 40 thicket phrases of the first thicket against the set of similar size for the second thicket. Taking into account a separate generalization of noun and verb phrases, this average case consists of 2* 45*45 generalizations, followed by the subsumption checks. Each phrase generalization is based on up to 12 string comparisons, taking an average size of phrase as 5 words. Hence on average the parse thicket generalization includes 2*45*45*12*5 operations. Since a string comparison takes a few microseconds, thicket generalization takes on average 100 milliseconds without use of index. However, in an industrial search application where phrases are stored in an inverse index, the generalization operation can be completed in constant time, irrespectively of the size of index (Lin, 2013). In case of map-reduce implementation (Lin&Dyer 2010) of generalization operation, for example, using Cascading framework, the time complexity becomes constant with the size of candidate search results to be re-ranked (Dean, 2009).

Acknowledgements

The author is grateful to Sergey Kuznetsov, Dmitry Ilvovsky, Fedor Strok, Boris Kovalerchuk, Daniel Usikov for the fruitful discussion and help in preparation of this manuscript.

References

1. Bhasker, B. K. Srikumar (2010). Recommender Systems in E-Commerce. CUP. ISBN 978-0-07-068067-8.
2. Bichindaritz, Isabelle, Sarada Akkineni. Concept mining for indexing medical literature. Engineering Applications of Artificial Intelligence, Volume 19, Issue 4, June 2006, pp 411–417. http://dx.doi.org/10.1016/j.engappai.2006.01.009.
3. Bron, Coen; Kerbosch, Joep (1973), Algorithm 457: finding all cliques of an undirected graph, Commun. ACM (ACM) 16 (9): 575–577.
4. Bunke, H. Graph-Based Tools for Data Mining and Machine Learning. Machine learning and data mining in pattern recognition. Lecture Notes in Computer Science, 2003, Volume 2734/2003, 7-19.
5. Bunke, H., P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento. A comparison of algorithms for maximum common subgraph on randomly connected graphs. Structural, Syntactic, and Statistical Pattern Recognition, pages 85–106, 2002.
6. Byun, H., Seong-Whan Lee. 2002. Applications of Support Vector Machines for Pattern Recognition: A Survey. In Proceedings of the First International Workshop on Pattern

Recognition with Support Vector Machines (SVM '02), Seong-Whan Lee and Alessandro Verri (Eds.). Springer-Verlag, London, UK, UK, 213-236.
7. Cascading en.wikipedia.org/wiki/Cascading. http://www.cascading.org/ 2013.
8. Collins, M., and Duffy, N. 2002. Convolution kernels for natural language. In Proceedings of NIPS, 625–632.
9. Conte, D. P. Foggia, and M. Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. Journal of Graph Algorithms and Applications, 11(1):99–143, 2007.
10. Conte, D. P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, No. 3 (2004) 265-298.
11. Dan Gusfield. 1997. Algorithms on Strings, Trees and Sequences. Cambridge University Press, Cambridge, UK.
12. Daniel Jurafsky, James H. Martin. Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2008.
13. Dean, Jeff. Challenges in Building Large-Scale Information Retrieval Systems. research.google.com/people/jeff/WSDM09-keynote.pdf 2009.
14. Domingos P. and Poon, H. Unsupervised Semantic Parsing, In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009. Singapore: ACL.
15. Ehrlich H.-C., Rarey M.: Maximum common subgraph isomorphism algorithms and their applications in molecular science: review. Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011, vol. 1 (1), pp. 68-79.
16. Finn, V.K. (1999) On the synthesis of cognitive procedures and the problem of induction. NTI Series 2, N1-2 pp. 8-45.
17. Fukunaga, K. Introduction to statistical pattern recognition (2nd ed.), Academic Press Professional, Inc., San Diego, CA, 1990.
18. Furukawa, K. (1998) From Deduction to Induction: Logical Perspective. The Logic Programming Paradigm. In Apt, K.R., Marek V.W., Truszczynski, M., Warren, D.S., Eds. Springer.
19. Galitsky, B. Natural Language Question Answering System: Technique of Semantic Headers. Advanced Knowledge International, Australia (2003).
20. Galitsky, B., Josep Lluis de la Rosa, Gábor Dobrocsi. Inferring the semantic properties of sentences by mining syntactic parse trees. Data & Knowledge Engineering. Volume 81-82, November (2012a) 21-45.
21. Galitsky, B., Daniel Usikov, Sergei O. Kuznetsov: Parse Thicket Representations for Answering Multi-sentence questions. 20th International Conference on Conceptual Structures, ICCS 2013 (2013).
22. Galitsky, B., Kuznetsov S, Learning communicative actions of conflicting human agents. J. Exp. Theor. Artif. Intell. 20(4): 277-317 (2008).
23. Galitsky, B., G. Dobrocsi, J.L. de la Rosa, Kuznetsov, S.O.: From Generalization of Syntactic Parse Trees to Conceptual Graphs, in M. Croitoru, S. Ferré, D. Lukose (Eds.): Conceptual Structures: From Information to Intelligence, 18th International Conference on Conceptual Structures, ICCS 2010, Lecture Notes in Artificial Intelligence, vol. 6208, pp. 185-190.(2010)
24. Galitsky, B., Gabor Dobrocsi, Josep Lluís de la Rosa, Sergei O. Kuznetsov: Using Generalization of Syntactic Parse Trees for Taxonomy Capture on the Web. 19th International Conference on Conceptual Structures, ICCS 2011: 104-117 (2011).
25. Galitsky, B., Content Inversion for User Searches and Product Recommendation Systems and Methods. US Patent Application, eBay number 47088.80 (2013).
26. Galitsky, B., Machine Learning of Syntactic Parse Trees for Search and Classification of Text. Engineering Applications of Artificial Intelligence, 2012, http://dx.doi.org/10.1016/j.engappai.2012.09.017.
27. Galitsky, B., MP González, CI Chesñevar. A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. Decision Support Systems, 46 - 3, 717-729 (2009).
28. Ganter, B, Kuznetsov SO Pattern Structures and Their Projections. In: Conceptual Structures: Broadening the Base. Lecture Notes in Computer Science Volume 2120, 2001, pp 129-142.
29. Ganter, B. and Sergei O. Kuznetsov. Pattern Structures and Their Projections ICCS '01, 129-142, 2001.

30. Gildea. D. 2003. Loosely tree-based alignment for machine translation. In Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03), pp. 80–87, Sapporo, Japan.
31. Haussler, D. 1999. Convolution kernels on discrete structures.
32. Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4), 2013.
33. Hennig-Thurau, Thorsten, André Marchand, and Paul Marx. (2012), Can Automated Group Recommender Systems Help Consumers Make Better Choices? Journal of Marketing, 76 (5), 89-109.
34. Hensman, S. and Dunnion, J. Automatically building conceptual graphs using VerbNet and WordNet. 2004 International Symposium on information and Communication Technologies, Las Vegas, Nevada, June 16–18, 2004. ACM International Conference Proceeding Series, vol. 90. Trinity College Dublin, pp.115–120, 2004.
35. Iwashita, Motoi, Shinsuke Shimogawa, Ken Nishimatsu. Semantic analysis and classification method for customer enquiries in telecommunication services. Engineering Applications of Artificial Intelligence, Volume 24, Issue 8, December 2011, Pages 1521–1531 http://dx.doi.org/10.1016/j.engappai.2011.02.016.
36. Kalervo, Jarvelin, Jaana Kekalainen: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 2002.
37. Kapoor, S and H. Ramesh, "Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs," SIAM J. Computing, vol. 24, pp. 247-265, 1995.
38. Kann, V. On the Approximability of the Maximum Common Subgraph Problem. In (STACS '92), Alain Finkel and Matthias Jantzen (Eds.). Springer-Verlag, London, UK, UK, 377-388, 1992.
39. Karp, Richard M. (1972), "Reducibility among combinatorial problems", in Miller, R. E.; Thatcher, J. W., Complexity of Computer Computations, New York: Plenum, pp. 85–103.
40. Kim, Jung-Jae, Piotr Pezik and Dietrich Rebholz-Schuhmann. MedEvi: Retrieving textual evidence of relations between biomedical concepts from Medline. Bioinformatics. Volume 24, Issue 11 pp. 1410-1412. 2008.
41. Koch, I. Enumerating all connected maximal common subgraphs in two graphs. Theoretical Computer Science, 250(1-2):1–30, 2001.
42. Kohavi, Ron. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Artificial Intelligence IJCAI 1995.
43. Lehrer, Adrienne (1974), Semantic Fields and Lexical Structure, Amsterdam: Benjamins.
44. Lin, J. and Chris Dyer, Data-Intensive Text Processing with MapReduce. Morgan & Claypool Publishers, 2010.
45. Lin, J. Data-Intensive Text Processing with MapReduce. intool.github.io/MapReduceAlgorithms/MapReduce-book-final.pdf , 2013.
46. Le, H.T. and Abeysinghe, G. (2003) 'A Study to Improve the Efficiency of a Discourse Parsing System', in A. Gelbukh (ed.) Proceedings of 4th International Conference on Intelligent Text Processing and Computational Linguistics, Vol. 2588, pp. 101–14. Berlin:Springer.
47. Lehmann, Christian 1982, "Directions for interlinear morphemic translations." Folia Linguistica 16:199-224.
48. Mann, William C., Christian M. I. M. Matthiessen and Sandra A. Thompson (1992). Rhetorical Structure Theory and Text Analysis. Discourse Description: Diverse linguistic analyses of a fund-raising text. ed. by W. C. Mann and S. A. Thompson. Amsterdam, John Benjamins: 39-78.
49. Manning, C. and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999.
50. Marcu (2000) Rhetorical Parsing of Unrestricted Texts. Computational Linguistics V 2 N3.
51. Marcu, D. (1997) 'From Discourse Structures to Text Summaries', in I. Mani and M.Maybury (eds) Proceedings of ACL Workshop on Intelligent Scalable Text Summarization, pp. 82–8, Madrid, Spain.
52. Mill, J.S. (1843) A system of logic, ratiocinative and inductive. London.

53. Mitchell, T. (1997) Machine Learning. McGraw Hill.
54. Montaner, M.; Lopez, B.; de la Rosa, J. L. (June 2003). A Taxonomy of Recommender Agents on the Internet. Artificial Intelligence Review 19 (4): 285–330.
55. Moon, J. W and Moser, L. (1965), On cliques in graphs, Israel J. Math. 3: 23–28.
56. Moschitti,A. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
57. Mandow, L., J.L. Pérez de la Cruz. A heuristic search algorithm with lexicographic goals. Engineering Applications of Artificial Intelligence, Volume 14, Issue 6, December 2001, pp 751–762. http://dx.doi.org/10.1016/S0952-1976(02)00009-X.
58. OpenNLP 2013. apache.org/opennlp/documentation/manual/opennlp.htm.
59. Palmer, Martha. "Semlink: Linking PropBank, VerbNet and FrameNet." Proceedings of the Generative Lexicon Conference. Sept. 2009, Pisa, Italy: GenLex-09, 2009.
60. Plotkin, G.D. A note on inductive generalization. In B. Meltzer and D. Michie, editors, Machine Intelligence, volume 5, pages 153-163. Elsevier North-Holland, New York, 1970.
61. Polovina S. and John Heaton, "An Introduction to Conceptual Graphs," AI Expert, pp. 36-43, 1992.
62. Punyakanok, V., Roth, D., & Yih, W. (2004). Mapping dependencies trees: an application to question answering. In: Proceedings of AI & Math, Florida, USA.
63. Punyakanok, V.,Roth, D. and Yih, W. The Necessity of Syntactic Parsing for Semantic Role Labeling. IJCAI-05.
64. Robinson J.A. A machine-oriented logic based on the resolution principle. Journal of the Association for Computing Machinery, 12:23-41, 1965.
65. Richter, Michael M. The search for knowledge, contexts, and Case-Based Reasoning. Engineering Applications of Artificial Intelligence, Volume 22, Issue 1, February 2009, Pages 3–9. http://dx.doi.org/10.1016/j.engappai.2008.04.021
66. Kuznetsov, SO and M.V. Samokhin, Learning Closed Sets of Labeled Graphs for Chemical Applications. In: Proc. 15th Conference on Inductive Logic Programming (ILP 2005), Lecture Notes in Artificial Intelligence (Springer), Vol.3625, pp.190-208., 2005.
67. Schilder, F. (2002) 'Robust Discourse Parsing via Discourse Markers, Topicality and Position', Natural Language Engineering 8(2/3): 235–55.
68. Searle, John. 1969. Speech acts: An essay in the philosophy of language. Cambridge, England: Cambridge University.
69. Son, Jeong-Woo, Tae-Gil Noh, Hyun-Je Song, Seong-Bae Park. An application for plagiarized source code detection based on a parse tree kernel. Engineering Applications of Artificial Intelligence, Volume 26, Issue 8, September 2013, pp. 1911–1918. http://dx.doi.org/10.1016/j.engappai.2013.06.007.
70. Sowa JF, Eileen C. Way: Implementing a Semantic Interpreter Using Conceptual Graphs. IBM Journal of Research and Development 30(1): 57-69 (1986) .
71. Sowa JF, Information Processing in Mind and Machine. Reading, MA: Addison-Wesley Publ., 1984.
72. Steinberger, J. Poesio, M. Kabadjov, M.A. Ježek, K. Two uses of anaphora resolution in summarization. Information Processing & Management, Volume 43, Issue 6, November 2007, Pages 1663-1680.
73. Sun, J.; Zhang, M.; and Tan, C. Tree Sequence Kernel for Natural Language. AAAI-25, 2011.
74. Sun, J.; Zhang, M.; and Tan, C. Exploring syntactic structural features for sub-tree alignment using bilingual tree kernels. In Proceedings of ACL, 306–315, 2010.
75. Trias i Mansilla, A., JL de la Rosa i Esteva. Asknext: An Agent Protocol for Social Search. Information Sciences 190, 144–161. 2012.
76. Velásquez, Juan D., Luis E. Dujovne, Gaston L'Huillier. Extracting significant Website Key Objects: A Semantic Web mining approach. Engineering Applications of Artificial Intelligence, Volume 24, Issue 8, December 2011, pp 1532–1541. http://dx.doi.org/10.1016/j.engappai.2011.02.001.
77. Vicient, Carlos, David Sánchez, Antonio Moreno. An automatic approach for ontology-based feature extraction from heterogeneous textual resources. Engineering Applications of Artificial Intelligence, Volume 26, Issue 3, March 2013, Pages 1092–1106. http://dx.doi.org/10.1016/j.engappai.2012.08.002.

78. Vismara, P. and B. Valery. Finding Maximum Common Connected Subgraphs Using Clique Detection or Constraint Satisfaction Algorithms. Modelling, Computation and Optimization in Information Systems and Management Sciences, pages 358–368, 2008.
79. Wu, Jiangning , Zhaoguo Xuan and Donghua Pan, Enhancing text representation for classification tasks with semantic graph structures, International Journal of Innovative Computing, Information and Control (ICIC), Volume 7, Number 5(B).
80. Xiong, Hao and Haitao Mi and Yang Liu and Qun Liu. Forest-Based Semantic Role Labeling, in Proc AAAI 2010.
81. Xing, Du and Xie Li. Accomodating domain-independence—a new approach to the development of general natural language interfaces. Engineering Applications of Artificial Intelligence, Volume 5, Issue 2, March 1992, Pages 135–144.
82. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. IEEE Int. Conf. on Data Mining, ICDM'02, IEEE Computer Society (2002) 721–724.
83. Yang, Shih-Yao and Von-Wun Soo. Extract conceptual graphs from plain texts in patent claims.Engineering Applications of Artificial Intelligence, Volume 25, Issue 4, June 2012, Pages 874–887. http://dx.doi.org/10.1016/j.engappai.2011.11.006.
84. Zhang, M.; Che, W.; Zhou, G.; Aw, A.; Tan, C.; Liu, T.; and Li, S. 2008. Semantic role labeling using a grammar-driven convolution tree kernel. IEEE transactions on audio, speech, and language processing 16(7):1315–1329.