Information Sciences 329 (2016) 125–143

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Generalization of parse trees for iterative taxonomy learning

Boris A. Galitsky*

eBay Inc., San Jose, CA 95125, USA

ARTICLE INFO

Article history:
Received 9 January 2013
Revised 3 July 2015
Accepted 11 September 2015
Available online 21 September 2015

Keywords:
Learning taxonomy
Web mining
Learning constituency parse tree
Search relevance

ABSTRACT

We build a taxonomy of entities which is intended to improve the relevance of search engine in a vertical domain. The taxonomy construction process starts from the seed entities and mines the web for new entities associated with them. To form these new entities, machine learning of syntactic parse trees (their generalization) is applied to the search results for existing entities to form commonalities between them. These commonality expressions then form parameters of existing entities, and are turned into new entities at the next learning iteration.

Taxonomy and paragraph-level syntactic generalization are applied to relevance improvement in search and text similarity assessment. We conduct an evaluation of the search relevance improvement in vertical and horizontal domains and observe significant contribution of the learned taxonomy in the former, and a noticeable contribution of a hybrid system in the latter domain. We also perform industrial evaluation of taxonomy and syntactic generalization-based text relevance assessment and conclude that proposed algorithm for automated taxonomy learning is suitable for integration into industrial systems. Proposed algorithm is implemented as a part of Apache *OpenNLP.Similarity* project.

1. Introduction

Nowadays, in designing search engines and text relevance systems, it is hard to overestimate the role of taxonomies for improving their precisions, especially in vertical domains. Taxonomies, thesauri or concept hierarchies are a crucial component of many applications of Information Retrieval and Natural Language Processing (NLP). However, building, tuning and managing taxonomies and ontologies is rather costly since a lot of manual operations are required. A number of studies proposed automated building of taxonomies based on linguistic resources and/or statistical machine learning (Kerschberg et al. [24], Roth [39], Kozareva et al. [25], Sánchez and Moreno [40]).

A majority of current approaches to taxonomy-supported searches have not been deployed in industry due to the following:

- insufficient accuracy of resultant search, due to a limited coverage of a domain by a taxonomy,
- limited expressiveness in representations of queries of real users,
- inability to disambiguate queries relying on taxonomies,
- high cost associated with manual construction or editing of linguistic resources, and their limited adjustability.

In terms of code reusability, taxonomies remain very labor intensive. Transferring a search engine from one domain to another, the primary component to be rebuilt is a taxonomy.

* Tel.: +1 6502094601.
E-mail address: bgalitsky@hotmail.com

http://dx.doi.org/10.1016/j.ins.2015.09.008
0020-0255/

In this work we will take advantage of web mining based on search engine APIs, deep syntactic parsing as well as learning and matching tree representations of sentences and taxonomies. Proposed industrial-quality taxonomy learning algorithm is oriented at an improvement of vertical search relevance and will be evaluated in a number of search settings and domains. The main challenge in building taxonomy tree is to make it as deep as possible to incorporate longer chains of relationships [22], so that more specific (and more complicated) questions can be answered.

A number of currently available general-purpose and crowd-sourced resources such as DBPedia and Freebase assist with entity-related searches, but are insufficient to filter out irrelevant answers concerning multiple entities, certain action over an entity and a multitude of its parameters. A set of available vertical ontologies, such as genes, bioinformatics, entertainment are also helpful for entity-based searches in vertical domains, however their taxonomy trees are rather shallow, and their utility for filtering out irrelevant answers is rather limited.

A few studies have attempted to learn taxonomies on the basis of textual input (Perrin and Petry [32]). Several researchers explored taxonomic relations explicitly expressed in texts by pattern matching (Hearst [20], Poesio et al. [34]). One limitation of pattern matching is that it involves the predefined choice of semantic relations to be extracted. In this study, to improve the flexibility of keyword-based pattern matching, we use matching of parse trees, which is a higher level of abstraction than sequences of words. We extend the notion of syntactic contexts of (Lin [26]) from a partial case such as noun + modifier and dependency triple toward finding a parse sub-tree in a parse tree. Our approach also extends handling of internal structure of noun phrases used to find taxonomic relations (Buitelaar et al. [4]). Many researchers follow Harris' distributional hypothesis of correlation between semantic similarity of words or terms, and the extent to which they share similar syntactic contexts (Harris [19]).

The contribution of this study is an automated taxonomy building mechanism which is based on initial set of main entities (a seed) for given vertical knowledge domain. This seed is then automatically extended by mining of web documents' abstracts which include a "meaning" of a current taxonomy node. This node is further extended by entities which are the results of inductive learning of commonalities between these documents. These commonalities are extracted using an operation of syntactic generalization, which finds the common parts of syntactic parse trees of a set of documents, obtained for the current taxonomy node. Syntactic generalization has been extensively evaluated commercially to improve text relevance (Galitsky et al. [17], Galitsky et al. [13]), and in this study we also apply it *at the level of paragraphs* for automated building of taxonomies. In (Galitsky [9]) taxonomy construction was considered from the standpoint of transfer learning [35,41], and sentence-level generalization was used.

For industrial search engines, the value of semantically-enabling search engines via taxonomies and ontologies for improving search relevance has been appreciated (Heddon [21]). Once a taxonomy adequately covering all important entities in a vertical domain is available, it can be directly applied to filtering out irrelevant answers. What is worth exploring nowadays is how to apply a real-world taxonomy to search relevance improvement, where this taxonomy is not perfect since it was automatically compiled from the web.

Our taxonomy building algorithm is focused on search relevance improvement, unlike the majority of ontology mining methods which optimize the precision and recall of extracted relations. Therefore evaluation in this study will assess the algorithm performance in terms of search accuracy improvement. Hence we expect the search performance-driven taxonomy learning algorithm to outperform the ones focused on most, or most exact, relations. Our evaluation will be conducted in the vertical and horizontal searches, as well as in industrial environment of text similarity assessment.

We now proceed to a formal problem formulation for a taxonomy-supported search. Let us consider a search query Q and candidate answers $a_1, a_2,\ldots, a_n \in A$ that are produced by a component of a search engine. In this paper we build a relevance verification component based on a taxonomy: it takes Q, A and filters out irrelevant answers to retain a subset of A. We need a mechanism which would determine which keywords in Q must occur in answer a_i; otherwise this answer would be considered irrelevant. To do that, we need to have a hierarchy (ordered sets) of keywords T for a specific vertical domain, so that for Q we can determine "what it is about" in terms of these keywords X which must occur in the relevant answer (then Q is about X and a_i is about X). Selected domain keywords are organized in a taxonomy T which enforces relevant a_i to have keywords X extracted from Q.

This paper elaborates this approach and is organized as follows. We first define the relationships between Q and X. We then explore the relationships between Q, X and A and propose T as a way to enforce relevance and propose an online search algorithm. After that we propose the methodology for how T is constructed, followed by evaluation of search relevance improvement.

The industrial evaluation of a hybrid system reveals that the proposed algorithm is suitable for integration into industrial systems. The algorithm is implemented as a component of Apache OpenNLP project.

2. Improving search relevance by taxonomies

To answer a question, natural language or keyword-based, it is beneficial to 'understand' what this question is about [3,10]. In the sense of current paper this 'understanding' is a preferential treatment of keywords. A Q/A system needs to understand the topic of the questions, what it is about, which keywords are most important. For example, for a query '*sales tax*' one can say that it is about *tax*, but not about *sale*. We denote a relationship between a set of keywords for a question and its subset which expresses the topic of this question as

is-about (*set-of-keywords, subset-of-keyword*).

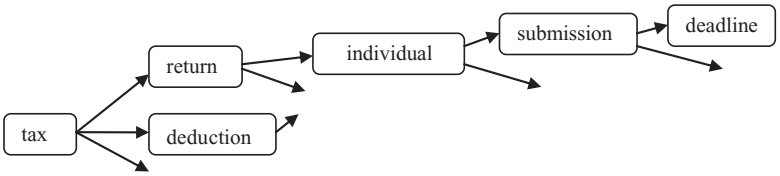

Fig. 1. A path of the taxonomy for a given query.

2.1. Must-occur keywords

For a query q with keywords {a b c} and its arbitrary relevant answer A we define that query is about b, *is-about(q, {b})* if queries {a b} and {b c} are relevant or marginally relevant to A, and {a c} is irrelevant to A. Our definition of query understanding, which is rather narrow, is the ability to say which keywords in the query are essential (such as b in the above example), so that without them the other query terms become meaningless. Also, an answer which does not contain b is irrelevant to the query which includes b.

For example, *is-about*({*machine, learning, algorithm*}, {*machine, learning*}), *is-about*({*machine, learning, algorithm*}, {*algorithm*}), *is-about*({*machine, learning, algorithm*}, {*learning, algorithm*}),

but not *is-about*({*machine, learning, algorithm*}, {*machine*}),

For query {a b c d}, if b is essential (*is-about*({a b c d}, {b}), c can also be essential when b is in the query such that {a b c}, {b c d}, {b c} are relevant, even {a b}, {b d} are (marginally) relevant, but {a d} is not (*is-about*({a b c d}, {b,c}). Hence for a query {a b c d} and two answers (snippets) {b c d … e f g} and {a c d … e f g}, the former is relevant and the latter is not.

A broader range of properties of how sets of keywords in a query correlate with sets of keywords in answers are formalized as operational semantics of default login in Galitsky [15,16].

In this paper we define taxonomies as means to systematically determine which keywords are essential in the query, which *must* and which *should* be in an answer, based on the above notion of query understanding via *is-about* relation. Taxonomies in the sense of this paper can be viewed as tree coding of a set of inter-connected *is-about* relations.

Notice that achieving relevancy using a taxonomy is based on totally different mechanism than a conventional TF*IDF based search. In the latter, importance of terms is based on the frequency of occurrence. For an NL query (not a Boolean query) any term can be omitted in the search result if the rest of terms give acceptable relevancy score. In case of a Boolean query, this is true for each of its conjunctive member. However, in the taxonomy-based search, according to the approach being introduced, we know which terms *should* occur in the answer and which terms *must* occur there, otherwise the search result becomes irrelevant.

2.2. Must-occur keywords in a taxonomy

Let us consider a totality of keywords in a domain D; these keywords occur in questions and answers in this domain. It turns out that there is always a hierarchy of these keywords: some are always more important than others in a sense of *is-about* relation. Keyword *tax* is more important than *deduction, individual, return, business*:

is-about({tax deduction}, {tax}) but not *is-about({tax deduction}, {deduction}*), since without context of *tax* keyword *deduction* is ambiguous;

is-about({individual, tax, return}, {tax, return}) but not *is-about({individual, tax, return}, {individual}),* since *individual* acquires sense as an adjective only in the context of *tax.*

At the same time, the above keywords are more important than partial cases of the situations they denote such as submission deadline:

is-about({individual, tax, return, submission, deadline}, {individual, tax, return}) but not *is-about({individual, tax, return, submission, deadline}, {submission, deadline})* because *submission deadline* may refer to something totally different (Fig. 1).

Hence *is-about* relation on subsets of D yields a partial order on certain subsets of D which we will encode via a tree T with the 'root' concept such as *tax* for the tree root. The tree T includes paths which correspond with typical queries.

We introduce a partial order on the set of subsets of keywords $K_1, K_2 \in 2^D$

$K_1 > K_2$ iff *is-about*($K_1 \cup K_2, K_1$) but not *is-about*($K_1 \cup K_2, K_2$).

We say that a path T_p covers a query Q if the set of keywords for the nodes of T_p is a super-set of Q. If multiple paths cover a query Q producing different intersections $Q \cap T_p$ then this query has multiple meanings in the domain; for each such meaning a separate set of acceptable answers is expected.

The search based on such tree structures is typically referred to as the *taxonomy-based search* (Galitsky [16]). It identifies terms that *should* occur in the answer and terms that *must* occur there, otherwise the search result is irrelevant. The keywords that should occur are from the taxonomy T, but the keywords that must occur are from both the query Q and taxonomy T. This is a totally different mechanism than a conventional TF*IDF based search.

2.3. Constructing relevance score function

Taxonomies can be applied in two ways:

(1) Accept/reject a given answer.
(2) Compute a similarity score induced by the taxonomy and then combine this score with other search scores such as TF*IDF relevance, popularity, geo-location and other product attributes.

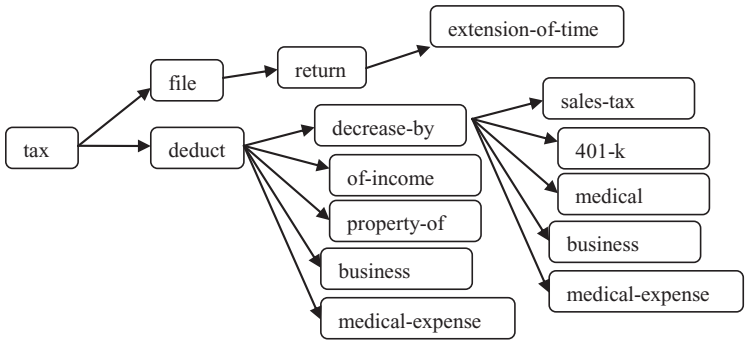

Fig. 2. A snapshot of the taxonomy for tax domain.

In this section we focus on the latter case. A similarity *score* function that measures the *similarity* between pairs <*Q, is-about(Q,X)*> and <a_i, *is-about*(a_i,X_i)>,

score (<*Q, is-about(Q,X)*>, <a_i, *is-about*(a_i,X_i)>) has a mapping to the interval [0,1]. Then we can re-rank answers a_i obtained by a conventional search engine, according to the score. The answers with the highest scores

$$\max_{i=1,2,\ldots,n} L(<Q, S(Q,X)>,\ <a_i, S(a_i,X_i)>)$$

are considered the *most relevant*.

The proposed measure of similarity takes into account the traditional approach, which takes all keywords from questions and answers, with our specific method of matching the keywords we determine as being essential. The above similarity will be assessed via mapping of both <*Q, is-about(Q,X)*> and <a_i, *is-about*(a_i,X_i)> into our taxonomy.

Let us consider a case where all must-occur words X from the query Q are also present in the answer, $X \subseteq X_i$ the answer a_i (taxonomy is not used here):

$$\text{If } X \subseteq X_i \text{ then score}(<Q,\ \textit{is-about}(Q,X)>,\ <a_i,\ \textit{is-about}(a_i,X_i)>) = 1.$$

However, this is a semantically shallow method especially for common situations where X and X_i contain only few words. A better way is to use the taxonomy to measure similarity between X and X_i:

$$\text{If } X \subseteq X_i \subseteq T \text{ then score}(<Q,\ \textit{is-about}(Q,X)>,\ <a_i,\ \textit{is-about}(a_i,X_i)>) = 1.$$

An answer $a_i \in A$ is *acceptable* if it includes *all must-occur* (according to *is-about*) keywords from the query Q as found in the taxonomy path $T_p \in T$. For any taxonomy path T_p which covers the question q (intersections of their keywords is not empty), these intersection keywords **must be** in the acceptable answer a_i.

$$\forall T_p \in T : T_p \cap X \neq \emptyset \Rightarrow X_i \supseteq T_p \cap X.$$

In other words, X as a set/tree of keywords for a question, which are essential in this domain (covered by a path in the taxonomy), must be a subset of X_i (the set/tree of keywords for this answer). This is a more complex (but not necessarily stronger) requirement than $X \subseteq X_i$. $\in$ used here to denote a path in a tree.

For the best answer (most accurate) we write

$$a_{best} : \max_i(|X_i \cap (T_p \cap X)|), \quad T_p \in T.$$

Accordingly, we define **a taxonomy-based relevance score** as the value of *cardinality* $|X_i \cap (T_p \cap X)|$, computed for all T_p which cover Q. Then the best answer is found among the scores for all answers A. The score can be normalized by dividing it by $|X|$ to get it in [0,1] interval.

The taxonomy-based score can be combined with the other scores such as TF*IDF, temporal/decay parameter, location distance, pricing, linguistic similarity, and other scores for the resultant ranking, depending on search engine architecture. In our evaluation we will be combining it with the linguistic similarity score. Hence the score is indeed depends not only on X and X_i but also on Q and a_i.

2.4. Examples of filtering answers based on taxonomy

To use the taxonomy to filter out irrelevant questions, we search for taxonomy path (down to a leaf node if possible) which is closest to the given question in terms of the number of entities from this question. Then this path and leave node specify most accurate meaning of the question, and constrain which entities *must* occur and which *should* occur in the answer to be considered relevant. If the nth node entity from the question occurs in answer, then all $k < n$ entities should occur in it as well.

Example 1. Consider a taxonomy T presented in Fig. 2 for the query Q = *'How can tax deduction be decreased by ignoring office expense'*, with a set of keywords *keywords(Q) = {how, can, tax, deduct(ion), decreas(ed)-by, ignor(ing), office, expense}* and a set of three answers $A = \{a1, a2, a3\}$ presented as keywords:

$$a1 = \{deduct,\ tax,\ business,\ expense,\ while,\ decreas(ing),\ holiday,\ travel,\ away,\ from,\ office\},$$

$$a2 = \{pay,\ decreas(ed),\ sales{-}tax,\ return,\ trip,\ from,\ office,\ to,\ holiday,\ no,\ deduct(ion)\},$$

$$a3 = \{when,\ file,\ tax,\ return,\ deduct,\ decrease{-}by,\ not,\ calculate,\ office,\ expense,\ and,\ employee,\ expense\}.$$

Notice that a2 includes the multiword sales-tax from the taxonomy, which is also counted as a set of two words {sales, tax}. However, in a3 *decrease-by* is considered as a single word because our scoring function considers prepositions as stop words and does not count them separately.

We show ending in brackets for convenience, omitting tokenization and word form normalization. Notice that in terms of keyword overlap, all of $a1$, $a2$ and $a3$ look like good answers, but it is not the case once we apply taxonomy.

In accordance with given T query Q is covered by the path T_p = *{<tax> - <deduct> - < decrease-by> - <office-expense>}*. We calculate the similarity score for each answer with Q:

$$score(a1) = cardinality(a1 \cap (Tp \cap Q)) = cardinality(\{tax, deduct\}) = 2;$$

$$score(a2) = cardinality(\{tax, deduct\}) = 2;$$

$$score(a3) = cardinality(\{tax, deduct, decrease{-}by, office{-}expense\}) = 3.$$

The answer a3 is the best answer in this example.

Our next example is about the disambiguation of keywords.

Example 2. Consider a *query q = 'When can I file extension of time for my tax return?'* with two answers:

a1 = 'You need to file form 1234 to request a 4 month extension of time to file your tax return'
a2 = 'You need to download file with extension 'pdf', print and complete it to do your taxes'
and the closest taxonomy path: T_p = *{<tax> - < file> - <return> - <extension-of-time>}*.

In this example both $a1$ and $a2$ contain word extension, but the keyword is "*extension-of-time*" not *extension*. Resolving this ambiguity leads to a higher score for $a1$.

Notice that there might be multiple nodes in a taxonomy tree, according to the way we build it. Another way to represent taxonomy is not to enforce it to be a tree (least general) but to allow only single node for each label instead (Fig. 3).

2.5. Taxonomy-based algorithm for filtering search results

We now outline the algorithm, which takes a query Q, runs a search (outside of this algorithm), gets a set of candidate answers A and finds the best acceptable answer according to the definitions given above.

The input: query Q, taxonomy T_p, the set of candidate answers A_a
The output: the best answer a_{best} and the set of acceptable answers A_a
We assume for query Q we obtained a set of candidate answers A by available means (using keywords, using internal index, or using external index of search engine's APIs);
(1) Find a path of taxonomy T_p which covers maximal number of keywords in Q, along with other paths, which cover Q, to form a set $P = \{T_{p1}, T_{p2}, \ldots\}$.
Unless acceptable answer is found:
(2) Compute the set $T_p \cap Q$.
For each answer $a_i \in A$
(3) Compute $a_i \cap (T_p \cap Q))$ and test if all essential words from the query, which exists in T_p are also in the answer (acceptability test)
(5) Compute similarity score of Q with or each a_i
(6) Compute the best answer a_{best} and the set of acceptable answers A_a.
If no acceptable answer found, return to (1) for the next path from P.
(7) Return a_{best} and the set of acceptable answers A_a if available.

This algorithm filters out irrelevant answers by searching for covering taxonomy path (down to a leaf node if possible) which is closest to the given query in terms of the number of keywords from this query. Then this path tells us about the topic of this query and the leaf node specifies the most accurate meaning of the query. If the nth node entity from the question occurs in answer, then all $k < n$ entities should occur in it as well.

3. Building taxonomies by web mining

Having understood how taxonomies work online to support search, we now focus on how to build them offline, having the whole richness of the web at our disposal.

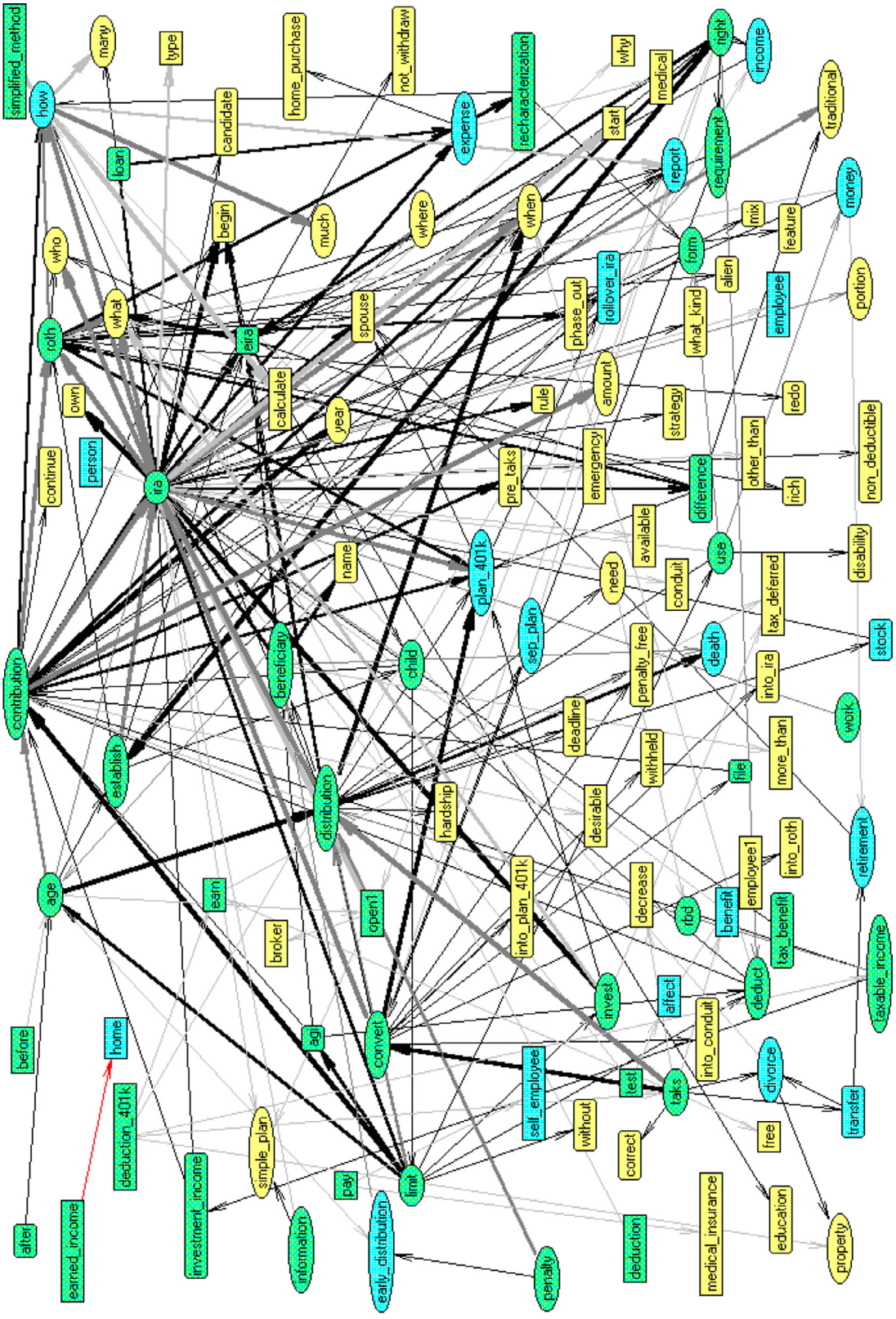

Fig. 3. Entity-centered taxonomy for Individual Retirement Arrangement domain.

3.1. Building taxonomy by generalizing search results

We build the taxonomies in an iterative manner. At a given step, we have certain entities in our current taxonomy, and we want to make these entities more precise by trying to find their parameters on the web. This approach is based on the observation that common expressions between search results for a given set of entities gives us *parameters* of these entities.

We grow tree T from the seed $T(0)$, which is the initial tree with the depth one to four. $T(0)$ enumerates the main entities of a given domain, and relations of these entities with a few domain-determining entities. For example, a seed for the tax domain will include the relationships

tax-deduct→tax-on-income→tax-on-property,

where *tax* is a domain-determining entity, and {*deduct, income, property*} are main entities in this domain. The objective of taxonomy learning is to acquire further parameters of existing entities such as *tax-deduct*. In the next iteration $T(1)$ of learning these parameters will be turned into entities (labels for the terminal nodes of $T(1)$, so that a new set of parameters will be learned to form $T(2)$ (from left to right, Fig. 2).

Each learning iteration is based on searching for the label of each current terminal node of $T(k)$. For the example above, we search for *tax-deduct, tax-on-income, tax-on-property* and extract words and expressions which occur in two or more search results. Common words are single verbs, nouns, adjectives and even adverbs or multi-words, including propositional, noun and verb phrases, which occur multiple times in distinct search result snippets. The next section explains how to extract common expressions between search results. After such common words and multi-words are identified, they are added as new entities to $T(k+1)$. For example, the path *tax-deduct* newly acquired edges of T can be

tax-deduct→decrease-by *tax-deduct→of-income*
tax-deduct→property-of *tax-deduct→business*
tax-deduct→medical-expense.

Now from the path in the taxonomy tree *tax-deduct* we obtained five new respective paths. The next step is to collect parameters for each path in the new set of leaves for $T(k+2)$. In our example, we run five queries and extract parameters for search results for each of them. The results will look like:

tax-deduct-decrease-by→sales
tax-deduct-decrease-by→401-K
tax-deduct-decrease-by→medical
tax-deduct-of-income→rental
tax-deduct-of-income→itemized
tax-deduct-of-income→mutual-funds

For example, searching the web for *tax-deduct-decrease* allows discovery of an entity *sales-tax* associated with *decrease of tax deduction*, usually with meaning 'sales tax'. Commonality between snippets shows that *sales tax* should be taken into account while calculating *tax deduction*, and not doing that would *decrease* it.

The taxonomy seed is formed manually or can be compiled from available domain-specific resources. Taxonomy seed can include, for example, a glossary of particular knowledge domain, readily available for most vertical domains.

We outline the recursive algorithm, which takes $T(k)$ and outputs $T(k+1)$. At the iteration k we acquire a set of nodes, extending current terminal node t_i with t_{ik1}, t_{ik2} …. This algorithm is based on the operation of generalization, which takes two snippets and outputs maximum common sub-parse tree (Section 4). We outline the iterative step:

Input: Taxonomy T_k with terminal nodes $\{t_1, t_2,\ldots, t_n\}$
A threshold for the number of occurrences to provide sufficient evidence for inclusion into T_k: $th(k, T)$.
Output: extended taxonomy T_{k+1} with terminal nodes
$\{t_{1k1}, t_{1k2},\ldots, t_{2k1}, t_{2k2},\ldots, t_{nk1}, t_{nk2}\}$
For each terminal node t_i
(1) Form a search query as a path from the root to t_i, $q = \{t_{root},\ldots,t_i\}$;
(2) Run web search for q and get a set of answers (snippets) A_q.
(3) Compute a pair-wise generalization (Section 4) for web search results W_q : $\cap(W_q) = w_1 \hat{} w_2, w_1 \hat{} w_3,\ldots, w_1 \hat{} w_m,\ldots,\ldots, w_{m-1} \hat{} w_m$,
(4) Arrange all elements of $\cap(W_q)$ in descending order of the number of occurrences in $\cap(W_q)$. Only keep the elements of $\cap(W_q)$ with the number of occurrences above a threshold $th(k, T)$. We call this set $\cap^{high}(W_q)$.
(5) Subtract the labels from all existing taxonomy nodes from $\Lambda^{high}(W_q)$:$\cap^{new}(W_q) = \cap^{high}(W_q)/T_k$. We maintain the uniqueness of labels of taxonomy to simplify the online matching algorithm.
(6) For each element of $\cap^{high}(W_q)$, create a taxonomy node t_{ihk}, where $h \in \cap^{high}(W_q)$, and k is the current iteration number, and add the taxonomy edge $(t_i, t_{ihk},)$ to T_k.
The default value of $th(k, T)$ is 2. However, there is an empirical limit on how many nodes are added to a given terminal node at each iteration. This limit is five nodes per iteration, so we take the five highest numbers of occurrences of a term in distinct search results. This constraint helps to maintain the tree topology for T. Given the algorithm for the iteration step, we apply it to the set of main entities at the first step, to build the whole taxonomy:
Input: Taxonomy T_o with nodes $\{t_1, t_2,\ldots, t_n\}$ which are main entities
Output: resultant taxonomy T with terminal nodes
Iterate through k:
Apply iterative step to k
If T_{k+1} has an empty set of nodes to add, stop

abandonment → [sale repossession; foreclosures reporting repossession; debt cancellation; repossession income foreclosure taxes; re-conveyance repossession; form repossession; consequences]

benefit→ [office child parent; credit child parent; support, child, parent; making child parent; resides child parent; taxpayer child parent; exclusion child parent; surviving benefits child, parent; reporting child parent]

hardship → [apply undue; taxpayer undue; irs undue; help undue; credits undue; cause undue; means required undue; court undue].

Fig. 4. Three sets of paths the tax domain.

3.2. Practical considerations

Fig. 4 shows the snapshot of taxonomy tree for three entities. For each entity, given the sequence of keywords, the reader can reconstruct the meaning in the context of *tax* domain. This snapshot illustrates the idea of taxonomy-based search relevance improvement: once the particular meaning (content, taxonomy path in our model) is established, we can find relevant answers. The head of the expression occurs in every path it yields.

It takes 20–40 h to construct a taxonomy for such complicated vertical domain as personal taxes or particular category of products, if the requests to a search engine APIs are sequential. Growth of parallel paths can occur in parallel using multithreading, which can reduce the taxonomy learning time by ten to thirty. The resultant tree includes 400–600 nodes and its paths are 4–8 nodes deep, the average depth being around six: relevance-based-on-parse-trees/src/test/resources/taxonomies.

Although we apply some frequency-based filtering building paths, the resultant taxonomy contains randomly obtained noisy paths, particularly due to spammed search results. However, these paths will very unlikely match both questions (assumed to be meaningful) and answers of reasonable quality. Hence noisy paths might insignificantly affect performance, but it is very unlikely that they would confirm an incorrect answer to a meaningful query.

4. Syntactic generalization of texts

To measure the similarity of logic formulas, a least-general generalization was proposed (Plotkin [33]). Given two terms, it produces a more general term that covers both, rather than a more specific one, as in unification.

To measure similarity between texts, we extend the notion of generalization from logic formulas to the sets of syntactic parse trees of these portions of text. Rather than extracting common keywords, the generalization operation produces a syntactic expression [12], maximal common sub-tree of parse trees for texts (sentences) [5]. This set of fragments of parse trees can be semantically interpreted as a common meaning shared by two sentences.

We present an example of the generalization operations of two sentences. The intermediate sub-trees are shown as lists for brevity. The generalization of distinct values is denoted by '*'. Let us consider the three following sentences:

I am curious how to use the digital zoom of this camera for filming insects.

How can I get short focus zoom lens for digital camera?

Can I get auto focus lens for digital camera?

We first draw the parse trees for these sentences and determine how to build their maximal common sub-trees (Fig. 5a):

We can see that the second and third trees are quite similar. Therefore, it is simple to build their common sub-tree as an (interrupted) path of the tree (Figs. 5(a) and 5(b)):

{*MD-can, PRP-I, VB-get, NN-focus, NN-lens, IN-for JJ-digital NN-camera*}. At the phrase level, we obtain:

`Noun phrases: [[NN-focus NN-*], [JJ-digital NN-camera]] Verb phrases: [[VB-get NN-focus NN-* NN-lens IN-for JJ-digital NN-camera]]`

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:

- Text
- Paragraph
- Sentence
- Phrases (noun, verb and others)
- Individual word

At each level except the lowest one, individual words, 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. Regarding the operations on trees, we follow the work of Kapoor and Ramesh [23].

When a text is a paragraph, the parse trees of sentences are combined to form the syntactic and discourse structure of this paragraph. We use the following relations to form the arcs between the nodes of parse trees for sentences:

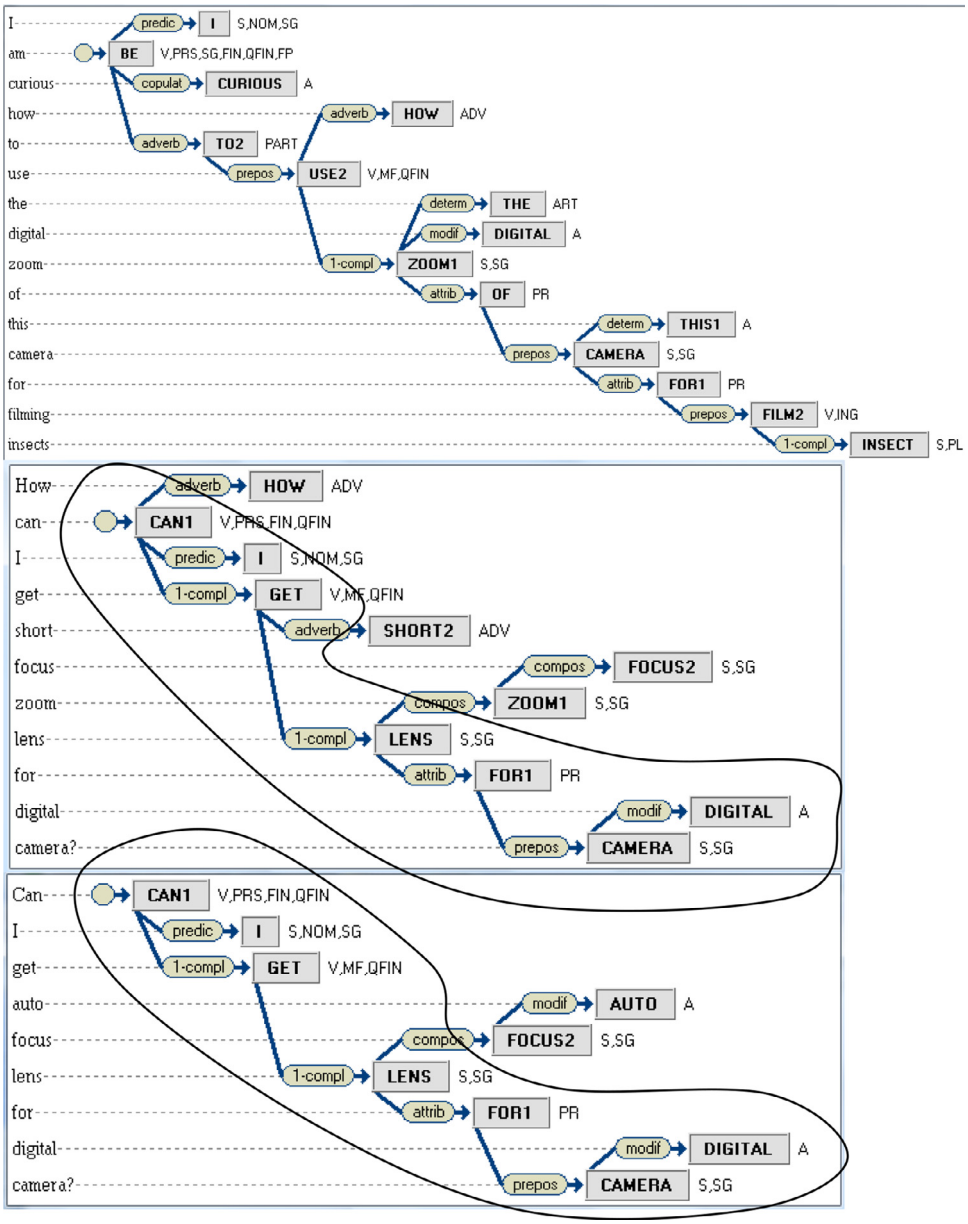

Fig. 5(a). Parse trees for three sentences. The curve shows the common sub-tree (in this case, there is only one) for the second and third sentences.

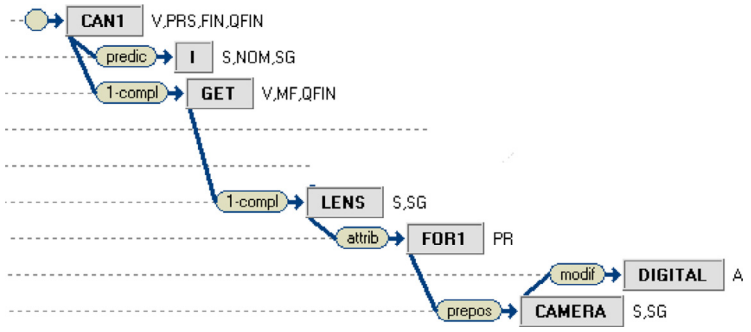

Fig. 5(b). Generalization results for the second and third sentences.

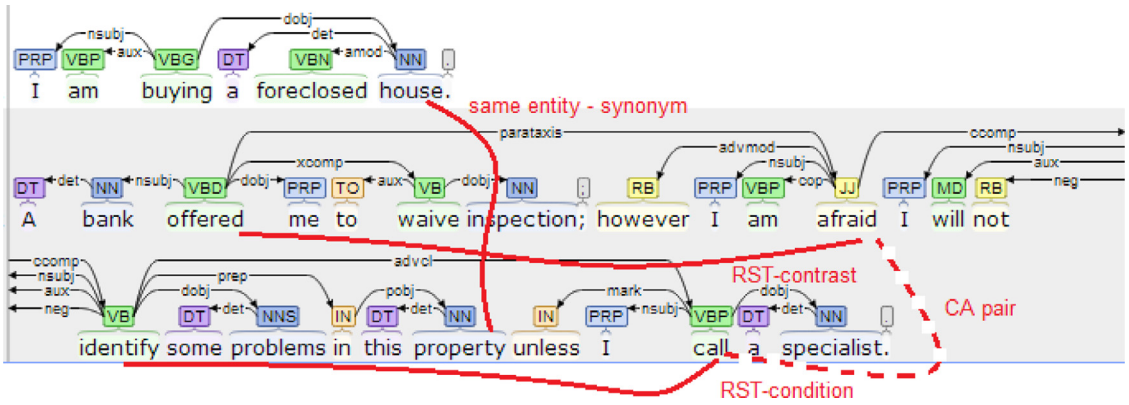

Fig. 6. Parse Thicket for a query.

- Anaphora
- Same entity/sub-entity and other relations between words which can obtained from external taxonomy
- Rhetoric relations
- Speech acts/communicative actions

Fig. 6 shows the parse thicket for a query which consists from three sentences. Stanford NLP software is used to obtain parse trees and anaphora, rhetoric relations and communicative actions are extracted by the in-house system. Further details about parse thickets are available at Galitsky [36].

4.1. Scoring of syntactic generalization

The default generalization score of a tree is a total number of nodes in all members of the set of maximal common sub-graphs. It is a measure of relevance between the parse thickets for the pair of texts subject to similarity assessment [43]. To make this measure more accurately reflecting the semantic similarity, we need to introduce weight for parts of speech, and also take into account the nature of parse thicket edges for syntactic relations as well as discourse relations.

We focused on adjustment the POS weights to deliver the most accurate similarity measure possible between sentences in Galitsky et al. [14], finding the optimal weights for nouns, adjectives, verbs and their forms so that the resultant search relevance is maximized. The search relevance was measured as a deviation in the ranking of search results obtained under current weight system, 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, whereas the other score weights were intact. This optimization gave us $W_{NN} = 1.0$, $W_{JJ} = 0.32$, $W_{RB} = 0.71$, $W_{CD} = 0.64$, $W_{VB} = 0.83$, 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 $W_{\langle POS,* \rangle} = 0.2$ for different words but the same POS, and $W_{\langle *,word \rangle} = 0.3$ for the same word occurring as different POSs within a sentence. Further details on optimizing the weights are available in Galitsky et al. [14].

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 score of the generalization of words $word_{sent_1}$ and $word_{sent_2}$:

$$score(sent_1 \wedge sent_2) = \sum_{\{NP,VP\}} \sum W_{POS} score, (word_{sent_1} \wedge word_{sent_2}).$$

Note that the internal sum ranges over words and the external sum ranges over phrases. 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.

5. Evaluation of search relevance improvement

Performing an evaluation of constructed taxonomy is frequently conducted manually by experts or researches themselves, or via comparison with existing ontologies like WordNet; also, some evaluation metrics have been defined for triplets produced by unsupervised web miner (Reinberger and Spyns [37]). In this study we perform evaluation of the resultant search engine leveraging the techniques introduced above instead of assessing the quality of acquired taxonomy directly. We start with an assessment for how syntactic generalization and taxonomies can improve search relevance in a set of vertical domain for various kinds of queries, and then proceed to the web (horizontal) search, where the algorithms of Sections 2 and 3 are implemented.

Table 1a
Improvement of accuracy in a vertical domain.

Query	Phrase sub-type	Relevance of baseline Yahoo search, %, averaging over 20 searches	Relevance of baseline Bing search, %, averaging over 20 searches	Relevance of baseline Google search, %, averaging over 20 searches	Relevance of re-sorting by parse thicket generalization, %, averaging over 20 searches	Relevance of re-sorting by using taxonomy, %, averaging over 20 searches	Relevance of re-sorting by using taxonomy and generalization, %, averaging over 20 searches	Relevance improvement for hybrid approach, comp. to baseline (averaged for Bing, Yahoo)
Short word phrases	Noun phrase	86.70	85.40	87.30	85.20	93.50	94.60	1.09
	Verb phrase	83.40	82.90	82.90	76.80	92.10	93.00	1.12
	How-to expression	76.70	78.20	79.10	80.30	93.40	94.50	1.21
	Average	82.27	82.17	83.10	80.77	93.00	94.03	1.14
Five–ten word phrases and sentences	Noun phrase	84.10	84.90	84.00	89.10	91.70	93.20	1.11
	Verb phrase	83.50	82.70	84.20	86.10	92.40	93.00	1.11
	How-to expression	82.00	82.90	81.60	83.00	88.90	91.90	1.12
	Average	83.20	83.50	83.27	86.07	91.00	92.70	1.11
Two–three sentences	One verb one noun phrases	68.80	67.60	60.20	68.90	81.20	84.30	1.29
	Both verb phrases	66.30	67.10	67.00	70.80	77.40	78.80	1.18
	One sent of how-to type	66.10	68.30	70.10	74.40	79.20	81.70	1.20
	Average	67.07	67.67	65.77	71.37	79.27	81.60	1.22

5.1. Evaluation settings of search relevance improvement

We conducted evaluation of relevance of taxonomy and syntactic generalization – enabled search engine, based on Yahoo, Bing and Google search engine APIs. For an individual query, the relevance was estimated as a percentage of correct hits among the first ten, using the values: {correct, marginally correct, incorrect} (this is along the lines of one of the measures in Resnik and Lin [38]). 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 20 search sessions. For our evaluation, we use the vertical domain of personal finance evaluated in Galitsky [16]. We also use customers' queries to eBay entertainment and product-related domains, from simple questions referring to a particular product, a particular user need, as well as a multi-sentence forum-style request to share a recommendation. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the authors.

To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (syntactic generalization score and taxonomy-based score). Google search results were scraped since no web search API is currently available. To evaluate the performance of a hybrid system, we used the weighted sum of these two scores (the weights were optimized in an earlier search sessions). For taxonomy-based approach, we followed the algorithm outlined in Section 2, and for syntactic generalization one – Section 4.

5.2. Vertical search

We start our evaluation in the same domain taxonomy learning was conducted (Table 1a). One can see that taxonomy contributes significantly in relevance improvement, compared to domain-independent syntactic generalization.

Notice that in a vertical domain where the taxonomy coverage is good (most questions are mapped well into taxonomy), paragraph-level syntactic generalization usually improves the relevance on its own, and as a part of hybrid system, however there are cases with no improvement. Taxonomy-based method is always helpful in a vertical domain, especially for a short queries (where most keywords are represented in the taxonomy) and multi-sentence queries (where the taxonomy helps to find the important keywords for matching with a question).

We can conclude for a vertical domain that a taxonomy should be definitely applied, and the syntactic generalization possibly applied, for improvement of relevance for all kinds of questions.

5.3. Web search relevance improvement

In a horizontal web search domain the contribution of taxonomy is about the same as that of paragraph-level syntactic generalization (Table 1b). Search relevance is improved by a 4–5% by a hybrid system, and is determined by a type of phrase (noun, verb) and query complexity. The longer and more complex the queries are, the higher relevance improvement is. Noun phrases perform better at the baseline and also in a hybrid system, than verb phrases and how-to phrases. Also note that the paragraph-level generalization can decrease relevance when applied to short queries, where linguistic information is not as important as TF*IDF analysis and the previous history of user clicks. Hybrid system almost always outperforms the individual components,

Table 1b
Evaluation of search relevance improvement in a horizontal domain.

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of baseline Google search, %, averaging over 20 searches	Relevancy of re-sorting by parse thicket generalization, %, averaging over 20 searches	Relevancy of re-sorting by using taxonomy, %, averaging over 20 searches	Relevancy of re-sorting by using taxonomy and generalization, %, averaging over 20 searches	Relevancy improvement for hybrid approach, comp. to baseline (averaged for Bing & Yahoo)
Three–four word phrases	Noun phrase	88.10	88.00	91.20	86.40	88.20	90.50	1.016
	Verb phrase	83.40	82.90	87.30	80.70	80.50	85.20	1.008
	How-to expression	76.70	81.20	81.00	77.40	77.00	80.40	1.010
	Average	82.73	84.03	86.50	81.50	81.90	85.37	1.011
Five–six word phrases	Noun phrase	86.30	85.40	84.90	88.50	85.80	88.40	1.034
	Verb phrase	84.40	85.20	84.40	87.30	88.30	88.70	1.048
	How-to expression	83.00	82.90	85.00	83.40	84.20	85.60	1.024
	Average	84.57	84.50	84.77	86.40	86.10	87.57	1.035
Seven–eight word phrases	Noun phrase	78.40	79.30	80.20	80.10	82.80	83.00	1.047
	Verb phrase	75.20	73.80	76.30	75.70	78.30	79.20	1.055
	How-to expression	73.20	73.90	74.00	73.80	77.80	76.30	1.035
	Average	75.60	75.67	76.83	76.53	79.63	79.50	1.046
Eight–ten word single sentences	Noun phrase	68.80	67.90	65.90	71.00	69.70	72.40	1.072
	Verb phrase	65.80	67.20	68.20	74.30	70.20	73.10	1.090
	How-to expression	64.30	63.90	64.10	66.50	67.50	68.10	1.062
	Average	66.30	66.33	66.07	70.60	69.13	71.20	1.075
Two sentences, >8 words total	One verb one noun phrases	66.50	67.20	68.40	66.50	69.20	70.20	1.042
	Both verb phrases	65.40	63.90	63.90	65.30	67.30	69.40	1.078
	One sent of how-to type	65.90	66.70	68.10	66.00	65.20	67.90	1.015
	Average	65.93	65.93	66.80	65.93	67.23	69.17	1.045
Three sentences, >12 words total	One verb one noun phrases	63.60	62.90	61.80	62.70	65.20	68.10	1.085
	Both verb phrases	63.10	64.70	65.20	64.50	62.50	67.20	1.045
	One sent of how-to type	64.20	65.30	64.30	65.00	64.70	66.80	1.034
	Average	63.63	64.30	63.77	64.07	64.13	67.37	1.055

so for a horizontal web search domain, paragraph-level syntactic generalization should definitely be used for multi-sentence queries, and the taxonomy-based search support is helpful for the selected questions covered by this taxonomy, and is useless for the majority of questions.

5.4. Taxonomy-supported search engine in news domain

We subject the proposed technique of taxonomy-based and syntactic generalization-based techniques to commercial mainstream news analysis at AllVoices.com (Fig. 7a). The task is to cluster relevant news items together by means of text relevance analysis. By definition, multiple news articles belong to the same cluster if there is a substantial overlap between the involved entities, such as geographical locations, the names of individuals, organizations and other agents, and the relationships between them. Some of these can be extracted using entity taggers and/or taxonomies built offline, and some are handled in real time using syntactic generalization (the bottom of Fig. 7b). The latter is applicable if there is a lack of prior entity information.

In addition to forming a cluster of relevant documents, syntactic generalization and taxonomy match was used to aggregate relevant images and videos from different sources, such as Google Image, YouTube and Flickr. It was implemented by assessing their relevance given their textual descriptions and tags.

The precision of the text analysis is achieved by the site's usability (click rate): more than 9 million unique visitors per month. Recall is accessed manually; however, the system needs to find at least a few articles, images and videos for each incoming article. Recall is generally not an issue for web mining and web document analysis (it is assumed that there is a sufficiently high number of articles, images and videos on the web for mining).

Relevance is ensured by two steps. First, we form a query to the image/video/blog search engine API, given an event title and first paragraph and extracting and filtering noun phrases by certain significance criteria. Second, we apply a similarity assessment to the texts returned from images/videos/blogs and ensure that substantial common noun, verb or prepositional sub-phrases can be identified between the seed events and these media (Fig. 7c).

The objective of syntactic generalization is to filter out false-positive relevance decisions made by a statistical relevance engines. This statistical engine has been designed following (Liu and Birnbaum [28,29]). The percentage of false-positive news stories was reduced from 29% to 17% (approximately 30,000 stories/month, viewed by 9 million unique users), and the percentage of false positive image attachment was reduced from 24% to 20% (approximately 3000 images and 500 videos attached to stories

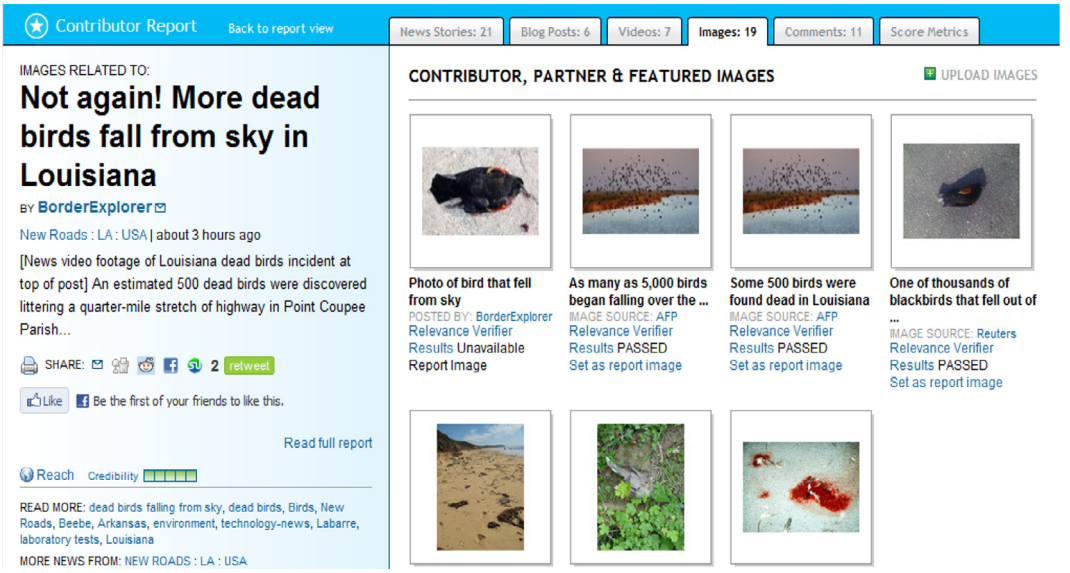

Fig. 7(a). News articles and aggregated images found on the web and determined to be relevant to this article.

Fig. 7(b). Syntactic generalization result for the seed articles and the other article mined for on the web.

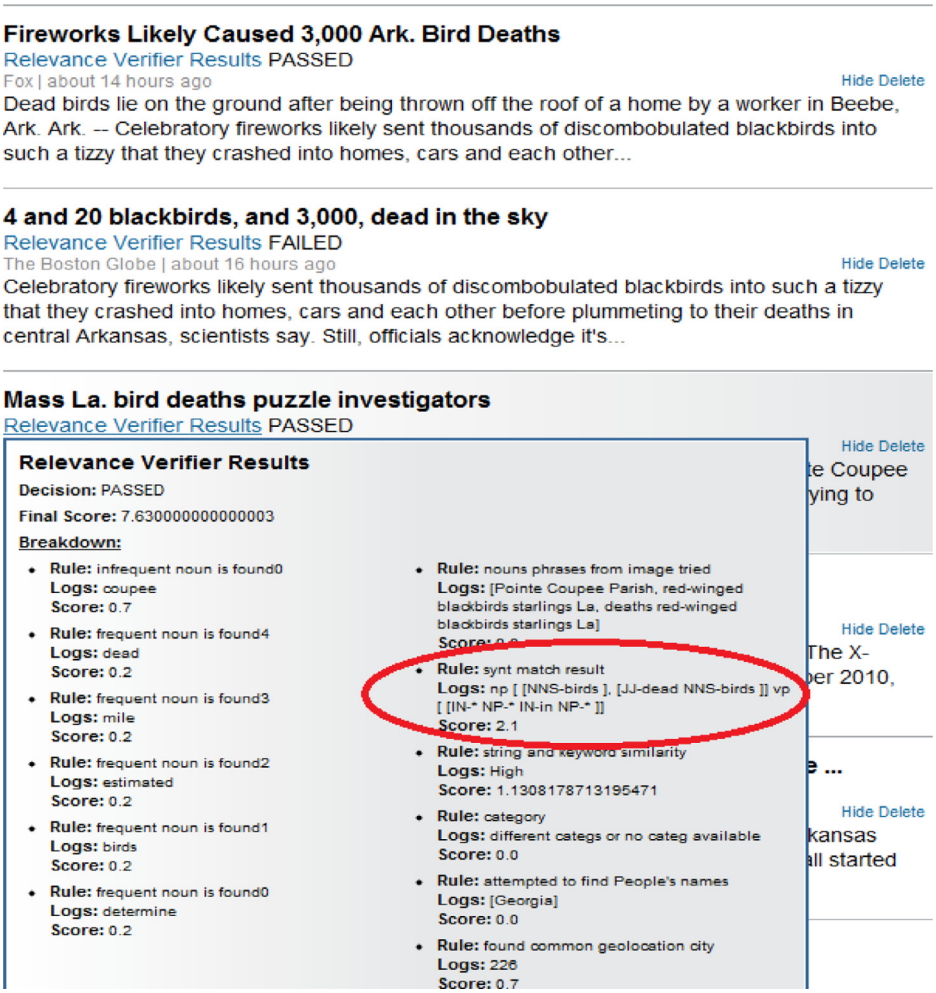

Fig. 7(c). Explanation for relevance decision while forming a cluster of news articles for the one in Fig. 7(a). The circled area shows the syntactic generalization result for the seed articles and the given one.

Table 2
Improving the error rate in the precision of text similarity.

Media/method of text similarity assessment	Full size news articles	Abstracts of articles	Blog posting	Comments	Images	Videos
Frequencies of terms in documents (baseline)	29.3%	26.1%	31.4%	32.0%	24.1%	25.2%
Syntactic generalization	19.7%	18.4%	20.8%	27.1%	20.1%	19.0%
Taxonomy-based	45.0%	41.7%	44.9%	52.3%	44.8%	43.1%
Hybrid syntactic generalization and taxonomy-based	17.2%	16.6%	17.5%	24.1%	20.2%	18.0%

monthly). The percentages shown are errors in the precision (100% – precision values); recall values are not as important for web mining, assuming there is an unlimited number of resources on the web and that we must identify the relevant ones.

The precision data for the relevance relationships between an article and other articles, blog postings, images and videos are presented in Table 2. Note that by itself, the taxonomy-based method has a very low precision and does not outperform the baseline of the statistical assessment. However, there is a noticeable improvement in the precision of the hybrid system, where the major contribution of syntactic generalization is improved by a few percentage points by the taxonomy-based method (Galitsky [9]). We can conclude that syntactic generalization and the taxonomy-based methods (which also rely on syntactic generalization) use different sources of relevance information. Therefore, they are complementary to each other.

The accuracy of our structural machine learning approach is worth comparing with the other parse tree learning approach based on the statistical learning of SVM. Moschitti [30] compares the performances of the bag-of-words kernel, syntactic parse trees and predicate argument structures kernel, and the semantic role kernel, confirming that the accuracy improves in this order and reaches an F-measure of 68% on the TREC dataset. Achieving comparable accuracies, the kernel-based approach requires manual adjustment; however, it does not provide similarity data in the explicit form of common sub-phrases. Structural

machine learning methods are better suited for performance-critical production environments serving hundreds millions of users because they better fit modern software quality assurance methodologies. Logs of the discovered commonality expressions are maintained and tracked, which ensures the required performance as the system evolves over time and the text classification domains change.

Unlike the current approach, which takes only syntactic level as a source, tree kernel-based methods also combine syntactic and semantic information (Zhou et al. [47]) to extract semantic relations between named entities. With a parse tree and an entity pair, a rich semantic relation tree structure is constructed to integrate both syntactic and semantic information [44]. Then it is subject to a context-sensitive convolution tree kernel, which enumerates both context-free and context-sensitive sub-trees by considering the paths of their ancestor nodes as their contexts to capture structural information in the tree structure.

6. Taxonomies for query expansion

In this section we will present taxonomies which make search more relevant not to a query but in terms of popularity of search results. This is important for a number of industrial product search and recommendation applications. In addition to verifying relevance of query keywords to product description, we attempt to recognize/predict user intention and make search results more specific.

For example, searching for '*female red shoes*', what do most people keep in mind? According to Google and heels.com, user intent here is most likely '*silhouette female red shoes*'. If a given search engine, attempting to acquire user intents from Google product search, has this kind of shoes in its index, it makes sense to put the respective products on the top. Typically, a smaller search engine for a product retailer does not have as rich data as Google, Bing. Amazon and eBay, but it can *learn* from major product search engines how to better rank search results by product popularity.

A special form of *Taxonomy for Query Expansion* helps in this task. We will describe a system for building this taxonomy. To acquire information on query popularity from the web, the system runs a given product search query at major product search engines, as well as against its own (internal) search index. Then the system compares search results, aggregated for the major search engines, and compares them with its own. For all products in the former set of search results, which are available in the system index, the system adds keywords from their description to the taxonomy during this learning process. Then online for an input query, the taxonomy serves as a look-up for query extension, adding the mined keywords and running extended query against the internal search index.

Hence this taxonomy is a map from the query string to extension string which we implemented via Lucene index. Offline, we select queries which deliver lower results (less relevant or less popular products), perform the mining session and add an entry to the taxonomy.

To evaluate the value of this taxonomy, we used the click-out rate (COR), a measure popular in industry to assess performance of search rather than search relevance itself. It turns out that for lower performing queries, the closer product search results to that of aggregated major search engines are, the higher COR is. Hence the objective of a smaller search engine is to recognize user intent as close to major search engines as possible. It can be achieved by the *Taxonomy for Query Expansion.*

We measure similarity of two search results as the score of their syntactic generalization.

Fig. 8 shows the overlap measure and normalized COR for 1100 queries ordered by the decrease of COR. Smooth curve is COR and oscillating curve is the overlap measure. Overlap measure shown is the generalization score (Section 4). One can see that COR

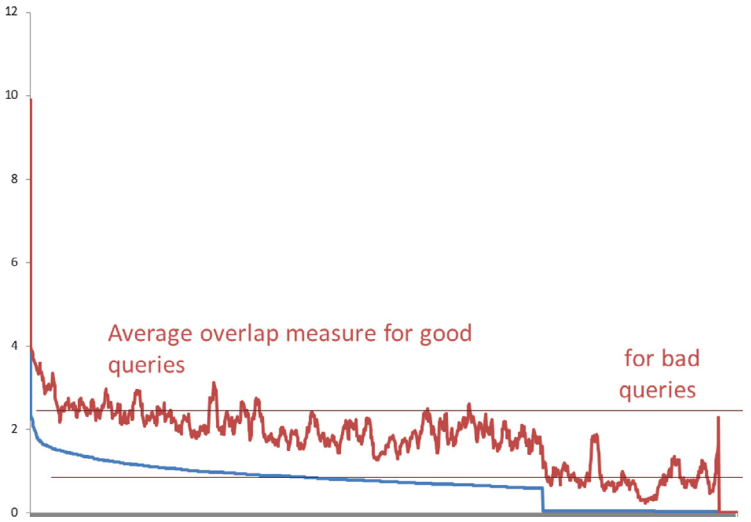

Fig. 8. The click-out rate values and search results overlap values for the sets of good (high click rate) and bad (low click rate) searches are sorted by the click rate in the descending order.

is correlated with the quality of search, as measured by the proximity to the search results aggregated through major product search engines. As relevance goes down (on average), so is the COR.

7. Using OpenNLP.Similarity.TaxonomyBuilder package

Taxonomy learner has been implemented as a part of *OpenNLP.Similarity* component for easy integration with search engines written in Java, especially Lucene SOLR/ElasticSearch based. Besides taxonomy learning and taxonomy-based support of search, *OpenNLP.Similarity* component performs a linguistic-based text relevance assessment, accepting two portions of texts (phrases, sentences, paragraphs) and returning a similarity score. Similarity component can be used to re-rank baseline search results to improve relevance, computing similarity score between a question and these results. Such score is based on both taxonomy match (Section 2) and syntactic generalization (Section 4). The code is available at

https://svn.apache.org/repos/asf/incubator/opennlp and also

http://code.google.com/p/relevance-based-on-parse-trees

The Apache OpenNLP library [31] is a machine learning based NLP toolkit. *OpenNLP.Similarity* component is an extension of OpenNLP intended for search and machine learning engineers to be easily integrated in Java-based production environments to perform relevance tasks. Besides search, *OpenNLP.Similarity* is useful for content generation, text clustering, web mining of images, videos, forums, blogs, and other media with textual descriptions. The objective of *OpenNLP.Similarity* component is to give a text-based application engineer a relevance tool which can be used as a black box: no understanding of computational linguistics or machine learning techniques are necessary.

7.1. Using search in Similarity component

To start with this component, please refer to
`SearchResultsProcessorTest.java` in package
`opennlp.tools.similarity.apps`

`public void testSearchOrder()` runs web search using Bing API and improves search relevance.

The reader is advised to consult

```
public List<HitBase> runSearch(String query)
```

and then at

```
private BingResponse calculateMatchScoreResortHits(BingResponse resp, String
searchQuery)
```

which gets search results from Bing and re-ranks them based on computed similarity score.

The main entry to Similarity component for short texts and sentences is

```
SentencePairMatchResult matchRes = sm.assessRelevance(snapshot, searchQuery);
```

where we pass the search query and the snapshot and obtain the similarity assessment structure which includes the similarity score.

To run this test you need to obtain search API key from Bing at
⟨10:monospace ⟩www.bing.com/developers/s/APIBasics.html⟨/10:monospace⟩ and specify it in
`public class BingQueryRunner in protected static final String APP_ID.`
For longer texts one should use package

`opennlp.tools.parse_thicket.matching` and function
`public List<List<ParseTreeChunk>> assessRelevance(String para1, String para2)` from class `Matcher`, which returns syntactic generalization results for two texts. The scoring function is then need to be applied to the generalization result to measure similarity between these texts: `package opennlp.tools.textsimilarity`, class `ParseTreeChunkListScorer` and function `public double getParseTreeChunkListScore (List<List< ParseTreeChunk>> matchResult)`.

7.2. Running taxonomy learner

To build a taxonomy for a domain, a knowledge engineer needs to compile a seed taxonomy. It can be formed based on a glossary of terms, domain introduction such as about.com or manually formed list. Also, an appropriate fragment from an arbitrary ontology or taxonomy for a broader domain can be used.

The following package is used to build taxonomies:

`opennlp.tools.similarity.apps.taxo_builder`

There are following components for building and running taxonomy:

`TaxonomyExtenderViaWebMining.java` performs web mining, by taking current taxonomy path, submitting formed keywords to Bing Azure API web search, obtaining snippets and possibly fragments of webpages, and extracting commonalities between them to add the next taxonomy node, as per Section 3.

`TaxoQuerySnapshotMatcher.java` is used in real time to obtain a taxonomy-based relevance score between a question and an answer.

`TaxonomySerializer.java` is used to write taxonomy in specified format: binary, text or XML. Taxonomy score, presented in Section 2, is calculated in function

```
public int getTaxoScore(String query, String snippet)
```

The resultant score can be combined with linguistic similarity score presented above, as has been evaluated in Section 5. For integration into SOLR search, taxonomy score can be included into class `SearchResultsReRankerRequestHandler` from package `opennlp.tools.similarity.apps.solr;`

`AriAdapter.java` is used to import seed taxonomy data from a Prolog ontology.

8. Related work and conclusions

Until recently, most approaches to the semantics of natural language were based on First Order Logic representations. Substantial development in NLP, the ability to acquire knowledge from the web and derive resources such as ontologies and taxonomies is expected to support more sophisticated semantic and pragmatics methods in search and other NLP tasks. A number of approaches are based on shallow representations of the text that rely on dependency structures and are mostly built to extend keyword matching (Durme et al. [11]). On the contrary, taxonomy learning in this work is based on full-scale dependency parsing and performed in a vertical domain, where ambiguity of terms is limited, and therefore fully automated taxonomy building and search support is possible. Also, the current work deals with syntactic tree transformation in the graph learning framework (compare with Chakrabarti and Faloutsos [8], Kapoor and Ramesh [23]) treating various phrasings for the same meaning in a more unified and automated manner. Hence our approach is finding a number of commercial applications including relevancy engine at citizens' journalism portal AllVoices.com, search and recommendation at Zvents.com and eBay.com.

Traditionally, semantic parsers are constructed manually, or are based on manually constructed semantic ontologies, but these are too delicate and costly. A number of supervised learning approaches to building taxonomic representations have been proposed (Cardie and Mooney [6], Reinberger and Spyns [37], Sánchez and Moreno [40], Wu [45]). Unsupervised approaches have been proposed as well, however they applied to shallow semantic tasks (Lin and Pantel [27]). The problem domain in the current study required much deeper handling of syntactic peculiarities to build taxonomies. In terms of learning, our approach is closer in merits to unsupervised learning of complete formal semantic representation. Compared to semantic role labeling (Carreras and Marquez [7]) and other forms of shallow semantic processing, our approach maps text to formal meaning representations, obtained via generalization.

Alani and Brewster [1] provide a means for re-use of existing taxonomies, based on the ranking of ontologies. This tool uses as input the search terms provided by a knowledge engineer and, using the output of an ontology text corpora is tokenized and syntactically analyzed before attribute extraction based on syntactic structures (Grefenstette [18], Reinberger and Spyns [37]).

We evaluated that the taxonomies, build from a wide variety of sources including blogs, forums, chats, opinion data, customer support data, are adequate to handle user queries searching for products and recommendations in vertical domains such as news, shopping and entertainment at AllVoices.com, Zvents.com, eBay.com, Become.com, as well as business and finance. Using taxonomies is just a single but necessary means of overall search relevance improvement; other means include user intention recognition (Zeng et al. [46]), learning from previous search session, and personalization. A number of metrics in an attempt to investigate the appropriateness of these means for ranking ontologies has been applied, and results were compared with a questionnaire-based human study. Also, fuzzy ontologies of Alexopoulos et al. [2] can be applied where the treatment and utilization of vague or imprecise knowledge are important. It might be suitable for a shopping domain like eBay, and to a lesser degree for the tax domain presented in this paper.

In this paper such quality of taxonomies as *appropriateness of concept hierarchies* is based on our trust in adequateness of search engines. Other studies such as Kuo and Lin [42] address this quality directly, propose a mechanism to identify the most suitable position to insert new terms into an existing concept hierarchy. The problem is challenging because there are hundreds or even thousands of candidate positions for insertion. Furthermore, usually there is no training instance available for an insertion; nor is it practical to assume the availability of a detailed description of the target concept, except in the hierarchy itself. To resolve the problem, Kuo and Lin [42] exploit the topology, content and social information, and apply a learning approach to identify the underlying construction criteria of the concept hierarchy, utilizing three metrics (namely, accuracy, taxonomic closeness, and ranking) to evaluate the proposed learning-based approach on a number of datasets to evaluate the proposed learning-based approach.

In this study we applied learning to syntactic parse trees in a systematic manner, taking advantage of a manifold of linguistic features beyond the bag of words. As we apply syntactic generalization to search results snippets, we extract the parameters of entities with much higher accuracy than a bag-of-words approach would do. Hence the quality of constructed taxonomy and the improved resultant search relevance is gained by full-scale linguistic processing. Since the size of taxonomy for a vertical domain is limited to thousands of words and multi-words, such linguistic processing is scalable.

We proposed a taxonomy building mechanism for a vertical domain, extending an approach where a taxonomy is formed based on specific semantic rules, specific semantic templates or a limited corpus of texts. Relying on web search engine API for taxonomy construction, we are leveraging not only the whole web universe of texts, but also the meanings formed by search engines as a result of learning from user search sessions. When a user selects certain search results, a web search engine acquires a set of associations between entities in questions and entities in answers. These associations are then used by our taxonomy learning process to find adequate parameters for entities being learned at a current taxonomy building step.

Two types of taxonomies have been proposed in this work:

- Taxonomy to increase search result *relevance to the query* by means of enforcing certain keywords from this query to occur in the search results;
- Taxonomy to increase search result *relevance to user intent* by learning it from other search engines and applying it via query expansion.

We evaluated the first taxonomy measuring search relevance and the second taxonomy in an industrial environment measuring normalized user click rate as a reflection of relevance and popularity of search results. To the best of our knowledge, it is the first algorithm deployed in industry which forms a taxonomy from the web in a fully automated manner.

We conclude that iterative taxonomy learning process with the support of syntactic generalization in both offline and online components is an effective way to use taxonomies in industrial environment. An improvement of relevance and click rate for product search by just a few percent for a set of tail (less popular) product gives a substantial increase in revenue. Complex queries for product searches usually indicate deep user knowledge and a serious intent to buy. Java-based *OpenNLP.Similarity* component serves as a good illustration of the proposed algorithm, and it is ready to be integrated with existing search engines such as SOLR and ElasticSearch.

References

[1] H. Alani, C. Brewster, Ontology ranking based on the analysis of concept structures, in: K-CAP '05 Proceedings of the 3rd International Conference on Knowledge Capture, 2005.
[2] Panos Alexopoulos, Manolis Wallace, Konstantinos Kafentzis, Dimitris Askouni, IKARUS-Onto: a methodology to develop fuzzy ontologies from crisp ones, Knowl. Inf. Syst. 32 (3) (2011) 1–29.
[3] J.F. Allen, Natural Language Understanding, Benjamin Cummings, 1987.
[4] P. Buitelaar, D. Olejnik, M. Sintek, A protégé plug-in for ontology extraction from text based on linguistic analysis, in: Proceedings of the International Semantic Web Conference (ISWC), 2003.
[5] H. Bunke, Graph-based tools for data mining and machine learning, Lecture Notes in Computer Science, vol. 2734, 2003, pp. 7–19.
[6] C. Cardie, R.J. Mooney, Machine learning and natural language, Mach. Learn. 1 (5) (1999) 5–9.
[7] X. Carreras, Luis Marquez, Introduction to the CoNLL-2004 shared task: semantic role labeling, in: Proceedings of the Eighth Conference on Computational Natural Language Learning, Boston, MA, ACL, 2004, pp. 89–97.
[8] D. Chakrabarti, C. Faloutsos, Graph mining: laws, generators, and algorithms, ACM Comput. Surv. 38 (1) (2006), doi:10.1145/1132952.1132954.
[9] B. Galitsky, Transfer learning of syntactic structures for building taxonomies for search engines, Eng. Appl. Artif. Intell. 26 (10) (November 2013) 2504–2515.
[10] David Sánchez, Jordi Castellà-Roca, Alexandre Viejo, Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines, Inf. Sci. 218 (2013) 17–30.
[11] B.V. Durme, Y. Huang, A. Kupsc, E. Nyberg, Towards light semantic processing for question answering, in: HLT Workshop on Text Meaning, 2003.
[12] M. Dzikovska, M. Swift, J. Allen, W. de Beaumont, Generic parsing for multi-domain semantic interpretation, in: International Workshop on Parsing Technologies (IWPT05), Vancouver BC, 2005.
[13] B. Galitsky, G. Dobrocsi, J.L. de la Rosa, S.O. Kuznetsov, Using generalization of syntactic parse trees for taxonomy capture on the web, in: ICCS, 2011, pp. 104–117.
[14] B. Galitsky, Josep Lluis de la Rosa, Gábor Dobrocsi, Inferring the semantic properties of sentences by mining syntactic parse trees, Data Knowl. Eng. 81–82 (November–December 2012) 21–45.
[15] B. Galitsky, Disambiguation via default rules under answering complex questions, Int. J. Artif. Intell. Tools 14 (1–2) (2005) 157, World Scientific.
[16] B. Galitsky, Natural Language Question Answering System: Technique of Semantic Headers (2003).
[17] B. Galitsky, G. Dobrocsi, J.L. de la Rosa, S.O. Kuznetsov, From generalization of syntactic parse trees to conceptual graphs, in: ICCS, 2010, pp. 185–190.
[18] G. Grefenstette, Explorations in Automatic Thesaurus Discovery, Kluwer Academic, 1994.
[19] Z. Harris, Mathematical Structures of Language, Wiley, 1968.
[20] M.A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: Proceedings of the 14th International Conference on Computational Linguistics, 1992, pp. 539–545.
[21] H. Heddon, Better living through taxonomies, Digital Web Mag. (2008). www.digital-web.com/articles/better_living_through_taxonomies/.
[22] R.W. Howard, Classifying types of concept and conceptual structure: some taxonomies, J. Cognit. Psychol. 4 (2) (April 1992) 81–111.
[23] S. Kapoor, H. Ramesh, Algorithms for enumerating all spanning trees of undirected and weighted graphs, SIAM J. Comput. 24 (1995) 247–265.
[24] L. Kerschberg, W. Kim, A. Scime, A semantic taxonomy-based personalizable meta-search agent, in: W. Truszkowski (Ed.), Innovative Concepts for Agent-Based Systems, vol. 2564: Lecture Notes in Artificial Intelligence, Springer-Verlag, Heidelberg, 2003, pp. 3–31.
[25] Z. Kozareva, Eduard Hovy, Ellen Riloff, Learning and evaluating the content and structure of a term taxonomy, in: Learning by Reading and Learning to Read AAAI Spring Symposium, Stanford, CA, 2009.
[26] D. Lin, Automatic retrieval and clustering of similar words, in: Proceedings of COLING-ACL98, 1998.
[27] D. Lin, P. Pantel, DIRT: discovery of inference rules from text, in: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2001, pp. 323–328.
[28] J. Liu, L. Birnbaum, What do they think? Aggregating local views about news events and topics, in: Proceedings of the 17th International Conference on World Wide Web (Beijing, China). WWW'08, ACM Press, New York, NY, 2008, pp. 1021–1022.
[29] J. Liu, L. Birnbaum, Measuring semantic similarity between named entities by searching the web directory, in: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, Washington, DC, USA, 2007, pp. 461–465, doi:10.1109/WI.2007.75.
[30] A. Moschitti, Efficient convolution kernels for dependency and constituent syntactic trees, in: Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
[31] OpenNLP apache.org/opennlp/, 2012.
[32] P. Perrin, Fred Petry, An information-theoretic based model for large-scale contextual text processing, Inf. Sci. 116 (2–4)) (1999) 229–252.
[33] G.D. Plotkin, A note on inductive generalization, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence, vol. 5, Edinburgh University Press, 1970, pp. 153–163.
[34] M. Poesio, T. Ishikawa, S. Schulte im Walde, R. Viera, Acquiring lexical knowledge for anaphora resolution, in: Proceedings of the 3rd Conference on Language Resources and Evaluation (LREC), 2002.
[35] R. Raina, A. Battle, H. Lee, B. Packer, A.Y. Ng, Self-taught learning: transfer learning from unlabeled data, in: Proceedings of 24th International Conference on Machine Learning, June 2007, pp. 759–766.
[36] B. Galitsky, Learning parse structure of paragraphs and its applications in search, Eng. Appl. Artif. Intell. 32 (2014) 160–184.
[37] M.-L. Reinberger, P. Spyns, Generating and evaluating triples for modeling a virtual environment, in: OTM Workshops, 2005, pp. 1205–1214.
[38] P. Resnik, J. Lin, Evaluation of NLP systems, in: A. Clark, C. Fox, S. Lappin (Eds.), The Handbook of Computational Linguistics and Natural Language Processing, Wiley-Blackwell, Oxford, UK, 2010.
[39] C. Roth, Compact, evolving community taxonomies using concept lattices, in: ICCS 14 – July 17–21, 2006, Aalborg, DK, 2006.

[40] D. Sánchez, A. Moreno, Web-scale taxonomy learning, in: ICML, 2005.
[41] S.J. Pan, Qiang Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[42] T.T. Kuo, S.-D. Lin, Learning-based concept-hierarchy refinement through exploiting topology, content and social information, Inf. Sci. 181 (12) (June 2011) 2512–2528.
[43] K. Wang, Z. Ming, T.-S. Chua, A syntactic tree matching approach to finding similar questions in community-based QA services, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09), New York, NY, USA, ACM, 2009, pp. 187–194.
[44] L. Wenyin, X. Quan, M. Feng, B. Qiu, A short text modeling method combining semantic and statistical information, Inf. Sci. 180 (20) (October 2010) 4031–4041.
[45] S.-T. Wu, Y. Li, Y. Xu, B. Pham, Y.-P. P. Chen, Automatic pattern-taxonomy extraction for web mining, in: IEEE/WIC International Conference on Web Intelligence (WI 2004), Beijing, China, September 20–24, 2004.
[46] Y. Zeng, N. Zhong, Y. Wang, Y. Qin, Z. Huang, H. Zhou, Y. Yao, F. van Harmelen, User-centric query refinement and processing using granularity-based strategies, Knowl. Inf. Syst. 27 (3) (2010) 419–450.
[47] Guodong Zhou, Longhua Qian, Jianxi Fan, Tree kernel-based semantic relation extraction with rich syntactic and semantic information, Inf. Sci. 180 (8) (April 2010) 1313–1325.