Scaling Web-based Acquisition of Entailment Relations

Idan Szpektor[◇] idan@szpektor.net
Hristo Tanev[†] tanev@itc.it
Ido Dagan[◦] dagan@cs.biu.ac.il
Bonaventura Coppola[†‡] coppolab@itc.it

[†]ITC-Irst, Via Sommarive, 18 (Povo) - 38050 Trento, Italy
[‡]DIT - University of Trento, Via Sommarive, 14 (Povo) - 38050 Trento, Italy
[◦]Department of Computer Science, Bar Ilan University - Ramat Gan 52900, Israel
[◇]Department of Computer Science, Tel Aviv University - Tel Aviv 69978, Israel

Abstract

Paraphrase recognition is a critical step for natural language interpretation. Accordingly, many NLP applications would benefit from high coverage knowledge bases of paraphrases. However, the scalability of state-of-the-art paraphrase acquisition approaches is still limited. We present a fully unsupervised learning algorithm for Web-based extraction of *entailment relations*, an extended model of paraphrases. We focus on increased scalability and generality with respect to prior work, eventually aiming at a full scale knowledge base. Our current implementation of the algorithm takes as its input a verb lexicon and for each verb searches the Web for related syntactic entailment templates. Experiments show promising results with respect to the ultimate goal, achieving much better scalability than prior Web-based methods.

1 Introduction

Modeling semantic variability in language has drawn a lot of attention in recent years. Many applications like QA, IR, IE and Machine Translation (Moldovan and Rus, 2001; Hermjakob et al., 2003; Jacquemin, 1999) have to recognize that the same meaning can be expressed in the text in a huge variety of surface forms. Substantial research has been dedicated to acquiring paraphrase patterns, which represent various forms in which a certain meaning can be expressed.

Following (Dagan and Glickman, 2004) we observe that a somewhat more general notion needed for applications is that of *entailment relations* (e.g. (Moldovan and Rus, 2001)). These are directional relations between two expressions, where the meaning of one can be entailed from the meaning of the other. For example *"X acquired Y"* entails *"X owns Y"*. These relations provide a broad framework for representing and recognizing semantic variability, as proposed in (Dagan and Glickman, 2004). For example, if a QA system has to answer the question *"Who owns Overture?"* and the corpus includes the phrase *"Yahoo acquired Overture"*, the system can use the known entailment relation to conclude that this phrase really indicates the desired answer. More examples of entailment relations, acquired by our method, can be found in Table 1 (section 4).

To perform such inferences at a broad scale, applications need to possess a large knowledge base (*KB*) of entailment patterns. We estimate such a *KB* should contain from between a handful to a few dozens of relations per meaning, which may sum to a few hundred thousands of relations for a broad domain, given that a typical lexicon includes tens of thousands of words.

Our research goal is to approach unsupervised acquisition of such a full scale *KB*. We focus on developing methods that acquire entailment relations from the Web, the largest available resource. To this end substantial improvements are needed in order to promote scalability relative to current Web-based approaches. In particular, we address two major goals: reducing dramatically the complexity of required auxiliary inputs, thus enabling to apply the methods at larger scales, and generalizing the types of structures that can be acquired. The algorithms described in this paper were applied for acquiring entailment relations for verb-based expressions. They successfully discovered several relations on average per each randomly selected expression.

2 Background and Motivations

This section provides a qualitative view of prior work, emphasizing the perspective of aiming at a full-scale paraphrase resource. As there are still no standard benchmarks, current quantitative results are not comparable in a consistent way.

The major idea in paraphrase acquisition is often to find linguistic structures, here termed *templates*, that share the same *anchors*. Anchors are lexical elements describing the context of a sentence. Templates that are extracted from different sentences and connect the same anchors in these sentences, are assumed to paraphrase each other. For example,

the sentences *"Yahoo bought Overture"* and *"Yahoo acquired Overture"* share the anchors {*X=Yahoo*, *Y=Overture*}, suggesting that the templates *'X buy Y'* and *'X acquire Y'* paraphrase each other. Algorithms for paraphrase acquisition address two problems: (a) finding matching anchors and (b) identifying template structure, as reviewed in the next two subsections.

2.1 Finding Matching Anchors

The prominent approach for paraphrase learning searches sentences that share common sets of multiple anchors, assuming they describe roughly the same fact or event. To facilitate finding many matching sentences, highly redundant comparable corpora have been used. These include multiple translations of the same text (Barzilay and McKeown, 2001) and corresponding articles from multiple news sources (Shinyama et al., 2002; Pang et al., 2003; Barzilay and Lee, 2003). While facilitating accuracy, we assume that comparable corpora cannot be a sole resource due to their limited availability.

Avoiding a comparable corpus, (Glickman and Dagan, 2003) developed statistical methods that match verb paraphrases within a regular corpus. Their limited scale results, obtaining several hundred verb paraphrases from a 15 million word corpus, suggest that much larger corpora are required.

Naturally, the largest available corpus is the Web. Since exhaustive processing of the Web is not feasible, (Duclaye et al., 2002) and (Ravichandran and Hovy, 2002) attempted bootstrapping approaches, which resemble the *mutual bootstrapping* method for Information Extraction of (Riloff and Jones, 1999). These methods start with a provided known set of anchors for a target meaning. For example, the known anchor set {*Mozart*, *1756*} is given as input in order to find paraphrases for the template *'X born in Y'*. Web searching is then used to find occurrences of the input anchor set, resulting in new templates that are supposed to specify the same relation as the original one ("born in"). These new templates are then exploited to get new anchor sets, which are subsequently processed as the initial {*Mozart*, *1756*}. Eventually, the overall procedure results in an iterative process able to induce templates from anchor sets and vice versa.

The limitation of this approach is the requirement for one input anchor set per target meaning. Preparing such input for all possible meanings in broad domains would be a huge task. As will be explained below, our method avoids this limitation by finding all anchor sets automatically in an unsupervised manner.

Finally, (Lin and Pantel, 2001) present a notably different approach that relies on matching separately *single* anchors. They limit the allowed structure of templates only to paths in dependency parses connecting two anchors. The algorithm constructs for each possible template two feature vectors, representing its co-occurrence statistics with the two anchors. Two templates with similar vectors are suggested as paraphrases (termed *inference rule*).

Matching of *single* anchors relies on the general distributional similarity principle and unlike the other methods does not require redundancy of sets of *multiple* anchors. Consequently, a much larger number of paraphrases can be found in a regular corpus. Lin and Pantel report experiments for 9 templates, in which their system extracted 10 correct inference rules on average per input template, from 1GB of news data. Yet, this method also suffers from certain limitations: (a) it identifies only templates with pre-specified structures; (b) accuracy seems more limited, due to the weaker notion of similarity; and (c) coverage is limited to the scope of an available corpus.

To conclude, several approaches exhaustively process different types of corpora, obtaining varying scales of output. On the other hand, the Web is a huge promising resource, but current Web-based methods suffer serious scalability constraints.

2.2 Identifying Template Structure

Paraphrasing approaches learn different kinds of template structures. Interesting algorithms are presented in (Pang et al., 2003; Barzilay and Lee, 2003). They learn linear patterns within similar contexts represented as finite state automata. Three classes of syntactic template learning approaches are presented in the literature: *learning of predicate argument templates* (Yangarber et al., 2000), *learning of syntactic chains* (Lin and Pantel, 2001) and *learning of sub-trees* (Sudo et al., 2003). The last approach is the most general with respect to the template form. However, its processing time increases exponentially with the size of the templates.

As a conclusion, state of the art approaches still learn templates of limited form and size, thus restricting generality of the learning process.

3 The TE/ASE Acquisition Method

Motivated by prior experience, we identify two major goals for scaling Web-based acquisition of entailment relations: (a) Covering the broadest possible range of meanings, while requiring minimal input and (b) Keeping template structures as gen-

eral as possible. To address the first goal we require as input only a phrasal lexicon of the relevant domain (including single words and multiword expressions). Broad coverage lexicons are widely available or may be constructed using known term acquisition techniques, making it a feasible and scalable input requirement. We then aim to acquire entailment relations that include any of the lexicon's entries. The second goal is addressed by a novel algorithm for extracting the most general templates being justified by the data.

For each lexicon entry, denoted a *pivot*, our extraction method performs two phases: (a) extract promising anchor sets for that pivot (*ASE*, Section 3.1), and (b) from sentences containing the anchor sets, extract templates for which an entailment relation holds with the pivot (*TE*, Section 3.2). Examples for verb pivots are: *'acquire'*, *'fall to'*, *'prevent'*. We will use the pivot *'prevent'* for examples through this section.

Before presenting the acquisition method we first define its output. A *template* is a dependency parse-tree fragment, with variable slots at some tree nodes (*e.g.* *'X* $\overset{subj}{\leftarrow}$ *prevent* $\overset{obj}{\rightarrow}$ *Y'*). An entailment relation between two templates *T1* and *T2* holds if the meaning of *T2* can be inferred from the meaning of *T1* (or vice versa) in some contexts, but not necessarily all, under the same variable instantiation. For example, *'X* $\overset{subj}{\leftarrow}$ *prevent* $\overset{obj}{\rightarrow}$ *Y'* entails *'X* $\overset{subj}{\leftarrow}$ *reduce* $\overset{obj}{\rightarrow}$ *Y risk'* because the sentence *"aspirin reduces heart attack risk"* can be inferred from *"aspirin prevents a first heart attack"*. Our output consists of pairs of templates for which an *entailment* relation holds.

3.1 Anchor Set Extraction (ASE)

The goal of this phase is to find a substantial number of promising anchor sets for each pivot. A good anchor-set should satisfy a proper balance between specificity and generality. On one hand, an anchor set should correspond to a sufficiently specific setting, so that entailment would hold between its different occurrences. On the other hand, it should be sufficiently frequent to appear with different entailing templates.

Finding good anchor sets based on just the input pivot is a hard task. Most methods identify good repeated anchors "in retrospect", that is after processing a full corpus, while previous Web-based methods require at least one good anchor set as input. Given our minimal input, we needed refined criteria that identify a priori the relatively few promising anchor sets within a sample of pivot occurrences.

ASE ALGORITHM STEPS:

For each pivot *(a lexicon entry)*

1. *Create a* pivot template, T_p
2. *Construct a parsed* sample corpus S *for* T_p*:*
   - *(a) Retrieve an initial sample from the Web*
   - *(b) Identify associated phrases for the pivot*
   - *(c) Extend* S *using the associated phrases*
3. *Extract candidate* anchor sets *from* S*:*
   - *(a) Extract* slot anchors
   - *(b) Extract* context anchors
4. *Filter the candidate anchor sets:*
   - *(a) by absolute frequency*
   - *(b) by conditional pivot probability*

Figure 1: Outline of the *ASE* algorithm.

The *ASE* algorithm (presented in Figure 1) performs 4 main steps.

STEP (1) creates a complete template, called the *pivot template* and denoted T_p, for the input pivot, denoted P. Variable slots are added for the major types of syntactic relations that interact with P, based on its syntactic type. These slots enable us to later match T_p with other templates. For verbs, we add slots for a subject and for an object or a modifier (*e.g.* *'X* $\overset{subj}{\leftarrow}$ *prevent* $\overset{obj}{\rightarrow}$ *Y'*).

STEP (2) constructs a *sample corpus*, denoted S, for the pivot template. STEP (2.A) utilizes a Web search engine to initialize S by retrieving sentences containing P. The sentences are parsed by the MINIPAR dependency parser (Lin, 1998), keeping only sentences that contain the complete syntactic template T_p (with all the variables instantiated).

STEP (2.B) identifies phrases that are statistically associated with T_p in S. We test all noun-phrases in S , discarding phrases that are too common on the Web (absolute frequency higher than a threshold MAXPHRASEF), such as *"desire"*. Then we select the N phrases with highest $tf \cdot idf$ score[1]. These phrases have a strong collocation relationship with the pivot P and are likely to indicate topical (rather than anecdotal) occurrences of P. For example, the phrases *"patient"* and *"American Dental Association"*, which indicate contexts of preventing health problems, were selected for the pivot *'prevent'*. Fi-

[1]Here, $tf \cdot idf = freq_S(X) \cdot \log\left(\frac{N}{freq_W(X)}\right)$ where $freq_S(X)$ is the number of occurrences in S containing X, N is the total number of Web documents, and $freq_W(X)$ is the number of Web documents containing X.

nally, STEP (2.C) expands S by querying the Web with the both P and each of the associated phrases, adding the retrieved sentences to S as in step (2.a).

STEP (3) extracts candidate anchor sets for T_p. From each sentence in S we try to generate one candidate set, containing noun phrases whose Web frequency is lower than MAXPHRASEF. STEP (3.A) extracts *slot anchors* – phrases that instantiate the slot variables of T_p. Each anchor is marked with the corresponding slot. For example, the anchors $\{$*antibiotics* $\overset{subj}{\leftarrow}$, *miscarriage* $\overset{obj}{\leftarrow}\}$ were extracted from the sentence *"antibiotics in pregnancy prevent miscarriage"*.

STEP (3.B) tries to extend each candidate set with one additional *context anchor*, in order to improve its specificity. This anchor is chosen as the highest $tf \cdot idf$ scoring phrase in the sentence, if it exists. In the previous example, '*pregnancy*' is selected.

STEP (4) filters out bad candidate anchor sets by two different criteria. STEP (4.A) maintains only candidates with absolute Web frequency within a threshold range [MINSETF, MAXSETF], to guarantee an appropriate specificity-generality level. STEP (4.B) guarantees sufficient (directional) association between the candidate anchor set c and T_p, by estimating

$$Prob(T_p|c) \approx \frac{freq_W(P \cap c)}{freq_W(c)}$$

where $freq_W$ is Web frequency and P is the pivot. We maintain only candidates for which this probability falls within a threshold range [SETMINP, SETMAXP]. Higher probability often corresponds to a strong linguistic collocation between the candidate and T_p, without any semantic entailment. Lower probability indicates coincidental co-occurrence, without a consistent semantic relation.

The remaining candidates in S become the input anchor-sets for the template extraction phase, for example, $\{$*Aspirin* $\overset{subj}{\leftarrow}$, *heart attack* $\overset{obj}{\leftarrow}\}$ for *'prevent'*.

3.2 Template Extraction (TE)

The Template Extraction algorithm accepts as its input a list of anchor sets extracted from *ASE* for each pivot template. Then, TE generates a set of syntactic templates which are supposed to maintain an entailment relationship with the initial pivot template. TE performs three main steps, described in the following subsections:

1. Acquisition of a sample corpus from the Web.
2. Extraction of *maximal most general templates* from that corpus.
3. Post-processing and final ranking of extracted templates.

3.2.1 Acquisition of a sample corpus from the Web

For each input anchor set, TE acquires from the Web a sample corpus of sentences containing it. For example, a sentence from the sample corpus for $\{$*aspirin, heart attack*$\}$ is: *"Aspirin stops heart attack?"*. All of the sample sentences are then parsed with MINIPAR (Lin, 1998), which generates from each sentence a syntactic directed acyclic graph *(DAG)* representing the dependency structure of the sentence. Each vertex in this graph is labeled with a word and some morphological information; each graph edge is labeled with the syntactic relation between the words it connects.

TE then substitutes each slot anchor (see section 3.1) in the parse graphs with its corresponding slot variable. Therefore, *"Aspirin stops heart attack?"* will be transformed into *'X stop Y'*. This way all the anchors for a certain slot are unified under the same variable name in all sentences. The parsed sentences related to all of the anchor sets are subsequently merged into a single set of parse graphs $S = \{P_1, P_2, \ldots, P_n\}$ (see P_1 and P_2 in Figure 2).

3.2.2 Extraction of maximal most general templates

The core of *TE* is a *General Structure Learning* algorithm (*GSL*) that is applied to the set of parse graphs S resulting from the previous step. *GSL* extracts single-rooted syntactic *DAG*s, which are named *spanning templates* since they must span at least over N_a slot variables, and should also appear in at least N_r sentences from S (In our experiments we set N_a=2 and N_r=2). *GSL* learns *maximal most general templates*: they are spanning templates which, at the same time, (a) cannot be generalized by further reduction and (b) cannot be further extended keeping the same generality level.

In order to properly define the notion of maximal most general templates, we introduce some formal definitions and notations.

DEFINITION: For a spanning template t we define a *sentence set*, denoted with $\sigma(t)$, as the set of all parsed sentences in S containing t.

For each pair of templates t_1 and t_2, we use the notation $t_1 \preceq t_2$ to denote that t_1 is included as a subgraph or is equal to t_2. We use the notation $t_1 \prec t_2$ when such inclusion holds strictly. We define $T(S)$ as the set of all spanning templates in the sample S.

DEFINITION: A spanning template $t \in T(S)$ is *maximal most general* if and only if both of the fol-

lowing conditions hold:

CONDITION A: For $\forall t' \in T(S), t' \preceq t$, it holds that $\sigma(t) = \sigma(t')$.

CONDITION B: For $\forall t' \in T(S), t \prec t'$, it holds that $\sigma(t) \supset \sigma(t')$.

Condition A ensures that the extracted templates do not contain spanning sub-structures that are more "general" (*i.e.* having a larger sentence set); condition B ensures that the template cannot be further enlarged without reducing its sentence set.

GSL performs template extraction in two main steps: (1) *build a compact graph representation* of all the parse graphs from S; (2) *extract templates* from the compact representation.

A compact graph representation is an *aggregate graph* which joins all the sentence graphs from S ensuring that all identical spanning sub-structures from different sentences are merged into a single one. Therefore, each vertex v (respectively, edge e) in the aggregate graph is either a copy of a corresponding vertex (edge) from a sentence graph P_i or it represents the merging of several identically labeled vertices (edges) from different sentences in S. The set of such sentences is defined as the *sentence set* of v (e), and is represented through the set of index numbers of related sentences (*e.g. "(1,2)"* in the third tree of Figure 2). We will denote with G_i the compact graph representation of the first i sentences in S. The parse trees P_1 and P_2 of two sentences and their related compact representation G_2 are shown in Figure 2.

Building the compact graph representation

The compact graph representation is built incrementally. The algorithm starts with an empty aggregate graph G_0 and then merges the sentence graphs from S one at a time into the aggregate structure.

Let's denote the current aggregate graph with $G_{i-1}(V_g, E_g)$ and let $P_i(V_p, E_p)$ be the parse graph which will be merged next. Note that the sentence set of P_i is a single element set $\{i\}$.

During each iteration a new graph is created as the union of both input graphs: $G_i = G_{i-1} \cup P_i$. Then, the following merging procedure is performed on the elements of G_i

1. ADDING GENERALIZED VERTICES TO G_i. For every two vertices $v_g \in V_g, v_p \in V_p$ having equal labels, a new *generalized vertex* v_g^{new} is created and added to G_i. The new vertex takes the same label and holds a sentence set which is formed from the sentence set of v_g by adding i to it. Still with reference to Figure 2, the generalized vertices in G_2 are '*X*', '*Y*' and '*stop*'. The algorithm connects the generalized vertex v_g^{new} with all the vertices which are connected with v_g and v_p.

2. MERGING EDGES. If two edges $e_g \in E_g$ and $e_p \in E_p$ have equal labels and their corresponding adjacent vertices have been merged, then e_a and e_p are also merged into a new edge. In Figure 2 the edges ('*stop*', '*X*') and ('*stop*', '*Y*') from P_1 and P_2 are eventually merged into G_2.

3. DELETING MERGED VERTICES. Every vertex v from V_p or V_g for which at least one generalized vertex v_g^{new} exists is deleted from G_i.

As an optimization step, we merge only vertices and edges that are included in equal spanning templates.

Extracting the templates

GSL extracts all maximal most general templates from the final compact representation G_n using the following sub-algorithm:

1. BUILDING MINIMAL SPANNING TREES. For every N_a different slot variables in G_n having a common ancestor, a minimal spanning tree st is built. Its sentence set is computed as the intersection of the sentence sets of its edges and vertices.

2. EXPANDING THE SPANNING TREES. Every minimal spanning tree st is expanded to the maximal sub-graph $maxst$ whose sentence set is equal to $\sigma(st)$. All maximal single-rooted *DAG*s in $maxst$ are extracted as candidate templates. Maximality ensures that the extracted templates cannot be expanded further while keeping the same sentence set, satisfying condition B.

3. FILTERING. Candidates which contain another candidate with a larger sentence set are filtered out. This step guarantees condition A.

In Figure 2 the maximal most general template in G_2 is '$X \overset{subj}{\leftarrow}$ *stop* $\overset{obj}{\rightarrow} Y$'.

3.2.3 Post-processing and ranking of extracted templates

As a last step, names and numbers are filtered out from the templates. Moreover, *TE* removes those templates which are very long or which appear with just one anchor set and in less than four sentences. Finally, the templates are sorted first by the number of anchor sets with which each template appeared, and then by the number of sentences in which they appeared.

4 Evaluation

We evaluated the results of the *TE/ASE* algorithm on a random lexicon of verbal forms and then assessed its performance on the extracted data through human-based judgments.

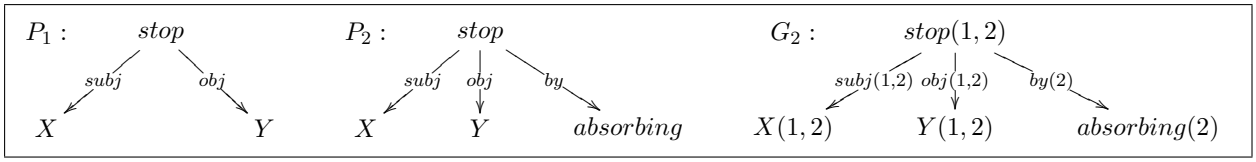

Figure 2: Two parse trees and their compact representation (sentence sets are shown in parentheses).

4.1 Experimental Setting

The test set for human evaluation was generated by picking out 53 random verbs from the 1000 most frequent ones found in a subset of the Reuters corpus[2]. For each verb entry in the lexicon, we provided the judges with the corresponding *pivot template* and the list of related candidate *entailment templates* found by the system. The judges were asked to evaluate entailment for a total of 752 templates, extracted for 53 pivot lexicon entries; Table 1 shows a sample of the evaluated templates; all of them are clearly good and were judged as correct ones.

Pivot Template	Entailment Templates
X prevent Y	X provides protection against Y X reduces Y X decreases the risk of Y X be cure for Y X a day keeps Y away X to combat Y
X accuse Y	X call Y indictable X testifies against Y Y defense before X
X acquire Y	X snap up Y Y shareholders approve X buyout Y shareholders receive shares of X stock
X go back to Y	Y allowed X to return

Table 1: Sample of templates found by *TE/ASE* and included in the evaluation test set.

Concerning the *ASE* algorithm, threshold parameters[3] were set as PHRASEMAXF=10^7, SETMINF=10^2, SETMAXF=10^5, SETMINP=0.066, and SETMAXP=0.666. An upper limit of 30 was imposed on the number of possible anchor sets used for each pivot. Since this last value turned out to be very conservative with respect to system coverage, we subsequently attempted to relax it to 50 (see Discussion in Section 4.3).

Further post-processing was necessary over extracted data in order to remove syntactic variations referring to the same candidate template (typically passive/active variations).

Three possible judgment categories have been considered: *Correct* if an entailment relationship *in at least one direction* holds between the judged template and the pivot template in some non-bizarre context; *Incorrect* if there is no reasonable context and variable instantiation in which entailment holds; *No Evaluation* if the judge cannot come to a definite conclusion.

4.2 Results

Each of the three assessors (referred to as $J_{\#1}$, $J_{\#2}$, and $J_{\#3}$) issued judgments for the 752 different templates. *Correct* templates resulted to be 283, 313, and 295 with respect to the three judges. *No evaluation*'s were 2, 0, and 16, while the remaining templates were judged *Incorrect*.

For each verb, we calculate *Yield* as the absolute number of Correct templates found and *Precision* as the percentage of good templates out of all extracted templates. Obtained Precision is 44.15%, averaged over the 53 verbs and the 3 judges. Considering *Low Majority* on judges, the precision value is 42.39%. Average Yield was 5.5 templates per verb.

These figures may be compared (informally, as data is incomparable) with average yield of 10.1 and average precision of 50.3% for the 9 "pivot" templates of (Lin and Pantel, 2001). The comparison suggests that it is possible to obtain from the (very noisy) web a similar range of precision as was obtained from a clean news corpus. It also indicates that there is potential for acquiring additional templates per pivot, which would require further research on broadening efficiently the search for additional web data per pivot.

Agreement among judges is measured by the *Kappa* value, which is 0.55 between $J_{\#1}$ and $J_{\#2}$, 0.57 between $J_{\#2}$ and $J_{\#3}$, and 0.63 between $J_{\#1}$ and $J_{\#3}$. Such Kappa values correspond to *moderate agreement* for the first two pairs and *substantial agreement* for the third one. In general, unanimous agreement among all of the three judges has been

[2]Known as Reuters Corpus, Volume 1, English Language, 1996-08-20 to 1997-08-19.

[3]All parameters were tuned on a disjoint development lexicon before the actual experiment.

reported on 519 out of 752 templates, which corresponds to 69%.

4.3 Discussion

Our algorithm obtained encouraging results, extracting a considerable amount of interesting templates and showing inherent capability of discovering complex semantic relations.

Concerning overall coverage, we managed to find correct templates for 86% of the verbs (46 out of 53). Nonetheless, presented results show a substantial margin of possible improvement. In fact yield values (5.5 *Low Majority*, up to 24 in best cases), which are our first concern, are inherently dependent on the breadth of Web search performed by the *ASE* algorithm. Due to computational time, the maximal number of anchor sets processed for each verb was held back to 30, significantly reducing the amount of retrieved data.

In order to further investigate *ASE* potential, we subsequently performed some extended experiment trials raising the number of anchor sets per pivot to 50. This time we randomly chose a subset of 10 verbs out of the less frequent ones in the original main experiment. Results for these verbs in the main experiment were an average Yield of 3 and an average Precision of 45.19%. In contrast, the extended experiments on these verbs achieved a 6.5 Yield and 59.95% Precision (average values). These results are indeed promising, and the substantial growth in Yield clearly indicates that the *TE/ASE* algorithms can be further improved. We thus suggest that the feasibility of our approach displays the inherent scalability of the *TE/ASE* process, and its potential to acquire a large entailment relation KB using a full scale lexicon.

A further improvement direction relates to template ranking and filtering. While in this paper we considered anchor sets to have equal weights, we are also carrying out experiments with weights based on cross-correlation between anchor sets.

5 Conclusions

We have described a scalable Web-based approach for entailment relation acquisition which requires only a standard phrasal lexicon as input. This minimal level of input is much simpler than required by earlier web-based approaches, while succeeding to maintain good performance. This result shows that it is possible to identify useful anchor sets in a fully unsupervised manner. The acquired templates demonstrate a broad range of semantic relations varying from synonymy to more complicated entailment. These templates go beyond trivial paraphrases, demonstrating the generality and viability of the presented approach.

From our current experiments we can expect to learn about 5 relations per lexicon entry, at least for the more frequent entries. Moreover, looking at the extended test, we can extrapolate a notably larger yield by broadening the search space. Together with the fact that we expect to find entailment relations for about 85% of a lexicon, it is a significant step towards scalability, indicating that we will be able to extract a large scale KB for a large scale lexicon.

In future work we aim to improve the yield by increasing the size of the sample-corpus in a qualitative way, as well as precision, using statistical methods such as supervised learning for better anchor set identification and cross-correlation between different pivots. We also plan to support noun phrases as input, in addition to verb phrases. Finally, we would like to extend the learning task to discover the correct entailment direction between acquired templates, completing the knowledge required by practical applications.

Like (Lin and Pantel, 2001), learning the context for which entailment relations are valid is beyond the scope of this paper. As stated, we learn entailment relations holding for some, but not necessarily all, contexts. In future work we also plan to find the valid contexts for entailment relations.

Acknowledgements

The authors would like to thank Oren Glickman (Bar Ilan University) for helpful discussions and assistance in the evaluation, Bernardo Magnini for his scientific supervision at ITC-irst, Alessandro Vallin and Danilo Giampiccolo (ITC-irst) for their help in developing the human based evaluation, and Prof. Yossi Matias (Tel-Aviv University) for supervising the first author. This work was partially supported by the MOREWEB project, financed by Provincia Autonoma di Trento. It was also partly carried out within the framework of the ITC-IRST (TRENTO, ITALY) – UNIVERSITY OF HAIFA (ISRAEL) collaboration project. For data visualization and analysis the authors intensively used the CLARK system (www.bultreebank.org) developed at the Bulgarian Academy of Sciences .

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL 2003*, pages 16–23, Edmonton, Canada.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL 2001*, pages 50–57, Toulose, France.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble.

Florence Duclaye, François Yvon, and Olivier Collin. 2002. Using the Web as a linguistic resource for learning reformulations automatically. In *Proceedings of LREC 2002*, pages 390–396, Las Palmas, Spain.

Oren Glickman and Ido Dagan. 2003. Identifying lexical paraphrases from a single corpus: a case study for verbs. In *Proceedings of RANLP 2003*.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2003. Natural language based reformulation resource and Web Exploitation. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of the 11th Text Retrieval Conference (TREC 2002)*, Gaithersburg, MD. NIST.

Christian Jacquemin. 1999. Syntagmatic and paradigmatic representations of term variation. In *Proceedings of ACL 1999*, pages 341–348.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for Question Answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC 1998*, Granada, Spain.

Dan Moldovan and Vasile Rus. 2001. Logic form transformation of WordNet and its applicability to Question Answering. In *Proceedings of ACL 2001*, pages 394–401, Toulose, France.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL 2003*, Edmonton, Canada.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a Question Answering system. In *Proceedings of ACL 2002*, Philadelphia, PA.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for Information Extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference (HLT 2002)*, San Diego, USA.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of ACL 2003*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for Information Extraction. In *Proceedings of COLING 2000*.