

Unsupervised Learning of Social Networks from a Multiple-Source News Corpus

Hristo Tanev
European Commission
Joint Research Centre
I21020 Ispra, Italy
hristo.tanev@jrc.it

Abstract

Social Networks provide an intuitive picture of inferred relationships between entities, such as people and organizations, which allows different analyst tasks to be performed. In this paper we describe an unsupervised syntax-based algorithm for learning of Social Networks from different news sources. The algorithm performs automatic paraphrase learning and multiple-source relation extraction. We put forward a novel syntactic graph matching algorithm which facilitates the method scalability. Finally, we demonstrate that automatically acquired Social Networks may be exploited successfully for certain analyst tasks.

Keywords social network, syntactic patterns, syntactic network, relation extraction, paraphrase learning

1. Introduction

Social Networks provide an intuitive picture of inferred relationships between entities, such as people and organizations; they allow for better understanding of the social systems and make possible application of a broad range of Social Network Analysis (SNA) approaches. SNA [11] focuses on a network-based view of the interrelations between entities to identify underlying groups, communication patterns, evolution of the relations [1], and other information.

However, manual building of Social Networks is feasible only on a very limited scale. For this reason, different authors describe automatic approaches for Social Network extraction (see among the others [4] and [5])

In this paper we present an unsupervised methodology for automatic learning of Social Networks from a multiple-source syntactically parsed news corpus and evaluate this approach against a real world data set. We compiled a English-language multi-source corpus of news articles using the Europe Media Monitor (EMM) family of applications [2]. In order to overcome the efficiency problems which emerge from using syntactic information on real-world data, we put forward an efficient graph matching algorithm.

This paper is structured as follows:

In section 2 we present related work. In section 3 we describe our method. In section 4 we describe the Syntactic Network model and a pattern matching algorithm based on it. In section 5 we present evaluation. Finally, we present our conclusions in section 6.

2. Related Work

Some approaches extract Social Networks from e-mail communications, FOAF links [5] and statistical co-occurrences [4]. A common disadvantage of these approaches is that they cannot detect the type of the extracted relations: For example, statistically derived relationship can reflect friendship, kinship, co-authorship, rivalry, etc. It is not possible to classify precisely the relations without analyzing deeper the context.

On the other hand, text-based Relation Extraction [12] provides more accurate means for Social Network extraction which allow for capturing a relation even from a single mention.

A predominant supervised paradigm for Relation Extraction is Support Vector Machines (SVM) [12]. In her PhD thesis Natasha Singh [8] showed that using syntactic features from a full parser brings significant improvement in the performance of SVM-based Relation Extraction. However, the use of syntactic features with SVM brings into light different efficiency problems: The SVM methods require each pair of names appearing in one sentence to be classified separately. This, combined with the time complexity of the syntax-based kernel functions [12] restricts the scalability of SVM methods based on syntactic features.

Another method for relation extraction based on full parsing is the syntactic pattern matching: [7] describes an unsupervised approach based on syntactic paraphrases. They apply each paraphrasing pattern against each tree from the corpus. This approach becomes quite inefficient when many patterns are matched against a big corpus. Moreover, their paraphrase acquisition based on [9] uses a huge quantity of Web search engine queries, which significantly slows down the learning process.

3. Unsupervised Learning of Social Networks

In contrast to SVM which requires manually labeled data to be provided, we use an unsupervised approach for learning of syntactic patterns which requires only a minimal human input. Our method is similar to the one described in [7]. However, our algorithm differs in several points: (i) We use news clusters instead of search engine queries for paraphrase learning. (ii) We use co-reference resolution (iii) We avoid matching each template to each sentence separately. We rather, perform efficient pattern matching which maps many templates to

many structures in a pretty much “all at once” manner. (iv) We learn simultaneously different relation types and use the information from one relation type when learning the others (see point 7 of the algorithm in section 3.1).

Our unsupervised social network acquisition method has three main stages:

1. Learning of syntactic patterns which paraphrase certain predefined relations
2. Extract relations from the news corpus
3. Aggregate the relations into a Social Network.

As a case study we used the “meeting” and “support” relations between two people.

Relation “meeting” holds between two people, if they met in a certain time interval. Relation “support” holds between two people, if one supports the other or have common opinions on important matters. More detailed description of our definition about these two relations is provided in the evaluation section.

EMM news clusters. For the paraphrase learning we used the daily EMM news clusters. Each news cluster consists of the news articles obtained from many news sources in one day whose main topic is the same. For example, the news about the meeting of the heads of state of Germany, France, and Russia are clustered into one news cluster.

3.1 Learning syntactic templates

The algorithm we put forward here is inspired by the TEASE approach described in [9]. Our learning schema includes the following main steps:

1. Provide manually a very small number of *seed* syntactic templates which express the main relation. For example, for the relation “X support Y” we use the syntactic patterns

$$X \leftarrow \text{subj} - \text{support} - \text{obj} \rightarrow Y \quad \text{and}$$

$$X \leftarrow \text{subj} - \text{praise} - \text{obj} \rightarrow Y \quad .$$
2. Match these templates against the news clusters in the corpus. Each pair of fillers of the slots X and Y is called an *anchor pair*. For example, if in the news the text “*Bush praised the Prime Minister Hamid Karzai*” appears, the algorithm will extract the anchor pair (*X: Bush; Y: Hamid Karzai*)
3. Normalize the anchor pairs using co-reference resolution and name variant detection; details of this step are discussed in section 3.3. After this step, the example anchor pair will become (*X: George W. Bush; Y: Hamid Karzai*).
4. For each extracted anchor pair, search in the same cluster all the sentences where both names of the anchor pair occur. The assumption is that the same relation will hold between the same pairs of names in the whole news cluster, since all articles in it have the same topic.
5. From all the sentences in which at least one anchor pair appears, learn syntactic templates using our

pattern-learning SyntNet GSL algorithm (see section 4.3).

6. Scoring: A template score is equal to the number of anchor pairs for which it appears.
7. Filtering: (i) Filter out all templates which appear for less than 2 anchor pairs. (ii) Accept all templates which are rooted in words which are also roots of highly scored patterns. (iii) Take out generic patterns like “X say Y”, “X have Y”, “X is Y”, etc. using a predefined template list (iv) Take out all templates which are included in templates with higher score in the other relation

The output of the learning algorithm is a set of syntactic templates which paraphrase the relation expressed by the seed templates or an entailment relation holds between them. For example “X meet with Y” paraphrases “X meet Y”. See *T1* and *T2* on Figure 1 as examples.

3.2 Extract relations from the news corpus

We match the patterns learned in the previous step against a syntactically parsed news corpus to extract pairs of people for which certain relation holds. We use an efficient pattern matching algorithm, whose details are described in section 4.

3.3 Information aggregation and co-reference resolution

We build two Social Networks – one for “meeting” and one for “support” relations. Each pair of people is connected via an arc in one of the Social Networks, if at least one instance of the corresponding relation is detected in the text. Co-reference resolution plays an important role during information aggregation. We used the EMM database in which each article identifier is associated with a set of names detected from a Named Entity recognizer. If we detect a partial name mention (e.g. *Bush*), we try to map it to some of the full names associated to the article (e.g. “*George W. Bush*”). We used also the EMM database of name variants to merge variants of the same name in one vertex in the Social Network. We did not perform pronominal anaphora resolution.

4. Syntactic Network and Efficient Algorithms for Template Learning and Matching

Our approach uses a news corpus parsed by a dependency parser, MiniPar [3].

The parser produces from each sentence a directed graph in which the nodes represent words and the arcs - syntactic relations between them (see *G1* and *G2* in Figure 1).

In order to make feasible efficient structure mining at syntactic level, we used the *Syntactic Network* model (*SyntNet* for short) which was presented earlier in [10]. It merges all the syntactic graphs in one big graph which

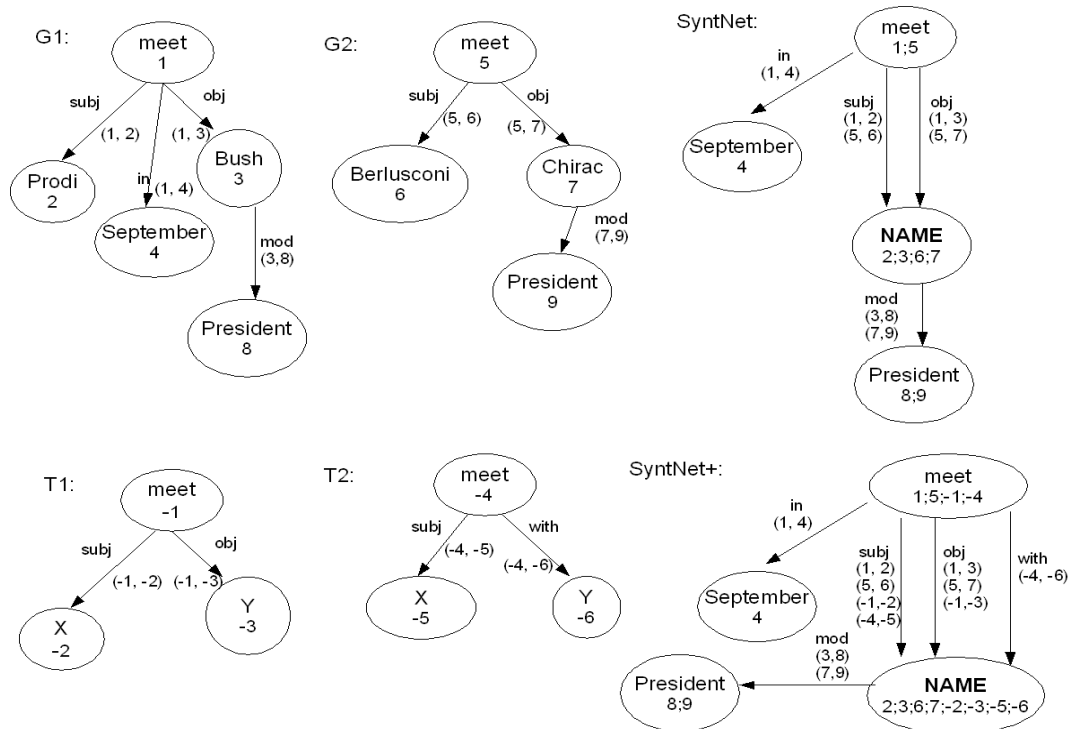


Figure 1: Merging sentences and templates in a Syntactic Network

allows for efficient searching of structures and calculation of their frequencies.

4.1 Structure of the SyntNet

Two dependency syntactic graphs $G1$ and $G2$ corresponding to the sentences “Prodi met President Bush in September” and “Berlusconi met President Chirac” are shown on Figure 1 together with their corresponding SyntNet.

Building the SyntNet passes through the following steps which will be illustrated using the example on Figure1:

1. All the vertices from the syntactic graphs obtain unique indices (see the numbers on the nodes of $G1$ and $G2$). Arcs obtain pair of indices which show which vertices they connect. For example, the arc *meet-subj-Prodi* in $G1$ is indexed with the index pair $(1, 2)$, because it connects vertices with indices 1 and 2.
2. All the vertices from the corpus (from $G1$ and $G2$ in this example) which have the same label and do not represent person names are merged in one vertex in SyntNet having the same label. For example, vertices number 1 and 5 have the label “meet” and they are merged in one aggregation vertex “meet” in SyntNet. To show which vertices are merged in the aggregation vertex, it is indexed with the indices of these vertices. For example, the vertex “meet” in SyntNet has two indices: 1 and 5, since it is obtained from the merging of vertices 1 and 5. The indices of one vertex in SyntNet form its *index set*.

3. All the person names from the whole corpus are merged in one vertex in SyntNet - see the vertex labeled “NAME”. This vertex is also indexed with the corresponding index set.
4. In a similar way, we merge all arcs which are labeled equally and which connect vertices which are merged. The index sets of the aggregation arcs are sets of index pairs which represent separate arcs. In SyntNet on Figure 1 the arc *meet-subj-NAME* has an index set $\{(1, 2); (5, 6)\}$, which shows that arcs *meet-subj-Prodi* - $(1, 2)$ and *meet-subj-Berlusconi* - $(5, 6)$ are merged in it.

A valuable property of the SyntNet is that isomorphic syntactic structures overlap, which allows for efficient tracing of common sub-structures. The worst case time complexity of building SyntNet is $O(|w| \log |w|)$, where $|w|$ is the number of the words in the parsed corpus.

4.2 Efficient matching of syntactic templates against SyntNet

Matching the templates against a corpus of parsed sentences is performed when we search for anchor pairs which fill the slots of the seed templates (see section 3.1) and during the relation extraction, when we extract pairs of related people from the corpus (see section 3.2). Taking into account that during our relation extraction experiments we had to match about 100 syntactic templates against more than a million of syntactic graphs, a straightforward approach of matching each template against each structure seemed to be quite time consuming.

We used the SyntNet graph model to substitute many to many matchings with one tracing of SyntNet: (i) First, we represent the corpus of parse graphs via a SyntNet, as it was described in the previous section. (ii) In a similar way, we add the syntactic templates to SyntNet and obtain SyntNet+ (iii) We trace SyntNet+ and using its indices, we find all paths in it which appear both in the corpus and in the templates. (iv) Finally, using the output from the previous step, we find the matching templates and the corresponding sub-structures in the corpus which they match.

Before continuing, we have to introduce some concepts. Each path in SyntNet, e.g. *meet-obj-NAME-mod-President* may represent a structure which has one or more occurrences in the corpus or which does not occur at all. If the path occurs in the corpus, its occurrences may be represented via their indices.

Definition 1: A path, represented by the indices of its vertices is called an *index path*.

Definition 2: If we have a corpus C and a SyntNet SN which represents it, each path p in a SN which occurs in C has a set of *corresponding index paths*, each representing one occurrence of p in C . The set of corresponding index paths of p is denoted by $ip(p)$.

As an example, consider the path $p_1 = \textit{meet-obj-NAME-mod-President}$ in SyntNet on Figure 1. Its $ip(p_1)$ contains two corresponding paths 1-obj-3-mod-8 and 5-obj-7-mod-9 . Each of them represents one occurrence of the path p_1 . On the other hand, the SyntNet path $p_2 = \textit{meet-subj-NAME-mod-President}$ has an empty set $ip(p_2)$, since it does not occur in the corpus.

We will illustrate the efficient template matching algorithm with an example in which we match a pair of templates ($T1: X \textit{meet} Y$ and $T2: X \textit{meet with} Y$) against the SyntNet on Figure 1.

1. We index the vertices of both templates with unique negative indices to differentiate from the positive indices of the corpus
2. We merge $T1$ and $T2$ with SyntNet. Slots X and Y are merged with the “ $NAME$ ” vertex. The result SyntNet we call SyntNet+ (see Figure 1)
3. In SyntNet+ we identify the vertices which represent template roots (only the vertex “ $meet$ ” in this example) and from these roots we traverse all this paths in SyntNet+ which have corresponding index paths with positive and negative indices. In this way we find these paths which appear both in the templates and in the corpus. We use a sub-algorithm called *Index Tracing* to accomplish this task, whose details will be explained later.

Let's consider the example on Figure 1: The Index Tracing will find two paths in SyntNet+ - *meet-subj-NAME* with four corresponding index paths: 1-subj-2 , 5-subj-6 , $(-1)\text{-subj-(-2)}$, and $(-4)\text{-subj-(-5)}$ and *meet-obj-NAME* with three

corresponding index paths: 1-obj-3 , 5-obj-7 , and $(-1)\text{-obj-(-3)}$. Each index path with negative indices refers to the templates $T1$ and $T2$ and each index path with positive indices refers to the corpus graphs $G1$ and $G2$. For example, the index path 1-subj-2 refers to the occurrence of *meet-subj-NAME* in $G1$, while $(-1)\text{-subj-(-2)}$ refers to the occurrence of the same path in the template $T1$. If a SyntNet+ path has corresponding index paths with positive and negative indices, as it is the case in this example, it occurs both in the templates and in the corpus.

4. Using all the SyntNet+ paths and corresponding index paths collected during the Index Tracing, we identify templates which completely match in the corpus and the vertices from the corpus where they match. This step of the algorithm is divided into several sub-steps:

4.1 Each template t is decomposed into set of paths $P(t)$, each of which has as an initial vertex the template root and its terminal vertex is a leaf. Following the example on Figure 1, $P(T1)$ consists of two paths - *meet-subj-X* and *meet-obj-Y* which are converted to *meet-subj-NAME* and *meet-obj-NAME*

4.2 If all the paths from $P(t)$ are found by Index Tracing, consider the template t for further processing, otherwise it can not be matched in the corpus.

4.3 If template t is to be considered, for each path p from $P(t)$, we consider the set of corresponding index paths $ip(p)$, found by the Index Tracing. For example, for the path $p = \textit{meet-subj-NAME}$ from $P(T1)$, $ip(p) = \{1\text{-subj-2}, 5\text{-subj-6}, (-1)\text{-subj-(-2)}, (-4)\text{-subj-(-5)}\}$.

4.4 For each path p from $P(t)$ we take its $ip(p)$ and form a set, denoted by $ini(p)$, which consists of all the initial vertices of the paths from $ip(p)$. For example, $ini(\textit{meet-sub-NAME}) = \{1, 5, -1, -4\}$. It is important to note that **the positive indices in $ini(p)$ refer to these vertices in the corpus from which an occurrence of the path p begins.**

4.5 For each template t under consideration, we find $rootset(t)$ as an intersection of the sets of initial indices:

$$rootset(t) = \bigcap_{p \in P(t)} ini(p)$$

For example:

$$rootset(T1) = ini(\textit{meet-subj-NAME}) \cap ini(\textit{meet-obj-NAME}) = \{1, 5, -1, -4\} \cap \{1, 5, -1\} = \{1, 5, -1\}$$

It is important to note that the positive indices in $rootset(t)$ refer to these vertices in the corpus in which occurrences of all the paths from $P(t)$ begin. **Therefore, each positive index in $rootset(t)$ refers to the root of one occurrence of the template t .**

5. For each template t and each root of its occurrence taken from $rootset(t)$, we may trace all the vertices of this occurrence. To do this, we use the corresponding index paths computed by the Index Tracing.
6. In each occurrence of t , a pair of names matches the slots of the template. We find all such name pairs from all the occurrences of t and return them as a final result. For each such a name pair, the semantic relation (e.g. “meeting”) expressed by t holds. In the example on Figure 1 the template T1 will match the pairs (“Prodi”, “Bush”) and (“Berlusconi”, “Chirac”).

Index Tracing. In its general form, this sub-algorithm takes on its input a SyntNet and a vertex v from the SyntNet, e.g. the vertex “meet” from SyntNet on Figure 1. The Index Tracing finds all the directed paths in a SyntNet representation for which this vertex v is initial¹. For each such a path p , the algorithm finds its corresponding index paths $ip(p)$. We will explain the Index Tracing basic steps using as an example SyntNet on Figure 1:

1. Initialization: $CurrentPathSet=\{v\}$. $ip(v)$ is initialized with the index set of v (see section 4.1), for example $ip(meet)=\{1,5\}$. The mapping between the single-vertex path v and $ip(v)$ is memorized in a hash table IP .
2. For each path p from $CurrentPathSet$, find an arc a from SyntNet, which can be added to p , such that a new SyntNet path pn is obtained.
As an example, let's assume $p=meet-obj-NAME$. If we take the arc $a=NAME-mod-President$, a new path can be formed $pn=meet-obj-NAME-mod-President$.
3. The set of corresponding index paths $ip(pn)$ is computed from $ip(p)$ by expanding some of the index paths from $ip(p)$ using the index set of the arc a . Index paths from $ip(p)$ which cannot be expanded are not included in $ip(pn)$.

Considering the values of p , pn , and a given in the example from the previous algorithm step, we can easily see that $ip(p)=\{1-obj-3, 5-obj-7\}$. The index path $1-obj-3$ can be expanded using the index pair $(3,8)$ from the index set of the arc a . The result will be a new index path $1-obj-3-mod-8$ which belongs to $ip(pn)$. In a similar way we expand the other index path from $ip(p)$ - $5-obj-7$ and obtain $5-obj-7-mod-9$. In this way we find that $ip(pn)=\{1-obj-3-mod-8, 5-obj-7-mod-9\}$.

¹ In the Index Tracing version used in SyntNet GSL, paths which finish in a certain vertex are built

4. If $ip(pn)$ is not empty, pn is added to $CurrentPathSet$ and $ip(pn)$ is memorized in a hash table IP , in which each path p_x is mapped to its $ip(p_x)$.
5. If expanding the paths from $CurrentPathSet$ is still possible, go to 2.
6. Finally, return $CurrentPathSet$ and the hash table IP .

4.3 SyntNet General Structure Learning (GSL)

GSL was described in [9]. From a set of anchor pairs and a parsed corpus it learns the most frequent and most general templates (see [9] for more formal description). We combined the idea of the GSL and the SyntNet and created a powerful and efficient algorithm for learning of general syntactic templates. SyntNet GSL uses the Index Tracing sub-algorithm to discover efficiently repeating sub-structures. Due to space limitations, we cannot give the details of this algorithm here.

4.4 Time complexity

If we assume that the templates are syntactic trees, but not imposing restrictions on the structure of parse graphs from the corpus, it can be shown that the worst case time complexity of the syntactic matching algorithm is bounded by $O((|s|+|t|) \log MaxArcO)^a$, where a is the maximal number of leaves of a template, $|s|$ is the number of the sentences in the corpus, $|t|$ is the number of the templates, and the $MaxArcO$ is the maximum number of occurrences of an SyntNet arc (e.g. *meet-obj-NAME*) in the corpus. When we use templates with two anchors, in most of the cases we have $a=2$. Considering this, the worst case time complexity becomes closer to $O((|s|+|t|) \log MaxArcO)$. Note that in general this estimate is better than $\Omega(|s| \cdot |t|)$, which is the time complexity estimate of each method which matches each template against each sentence. It can be shown also that the worst-case time complexity of SyntNet GSL is $O(|s| \cdot \log |s|)$.

5. Experiments and Evaluation

5.1 Evaluation schema

Social Networks provide higher level view of the relations between entities, such as people. They do not represent the individual events which motivate the relations. In our evaluation schema we assume that a relation “meeting” or “support” holds between two people X and Y, if at least one corresponding event took place in a certain time interval, the event involved both X and Y, and it is reflected in the news from the test corpus.

More formally, we consider the following definitions:

Definition 3: We say that “a meeting event” involves X and Y, if X and Y met physically and had some kind of important conversation.

Definition 4: We say that “a support event” involves X and Y, if one of the three event types took place: (i) X expressed support or positive attitude for Y or vice versa; (ii) X and Y reached some agreement with mutual benefits; (iii) X and Y have very similar attitudes towards important problems.

Our test corpus contains news articles from the period 03/Oct/2006 – 31/Oct/2006. It mostly covers events which took place in this period or shortly before or after that. We noticed that many news refer to events which happened the previous or the next week. Therefore, we decided to consider these events which are mentioned in the test corpus and which took place or at least overlap with the period which begins one week before 03/Oct/2006 and which finishes one week after in 31/Oct/2006.

Now we can define more formally “meeting” and “support” relations:

1. If at least one “meeting event” is taken into consideration and it involves X and Y, then we say that a “meeting relation” holds between X and Y.
2. If at least one “support event” is taken into consideration and it involves X and Y, then we say that a “support relation” holds between X and Y. For the purpose of the evaluation, we consider the “support” relation to be symmetric.

Note that in the Social Network many mentions of the same event and even the different events of the same type between the same pair of people map into the same relation. This is relevant especially in cases when the considered time interval is limited; in our case it is 28 days. In this clue, we evaluated the capability of our Social Network extraction algorithm to detect “meeting” and “support” relations. It was out of the scope of this evaluation to measure the performance of our system with respect to the detection of the individual events which motivate the relation.

5.2 Relation Extraction experiments

For paraphrase learning we used a training corpus of 98'000 English-language news articles clustered in 22'000 EMM topic clusters published in the period 01/May/2006 – 03/Oct/2006.

For testing the method, we used 125'000 English-language news articles published in the period 03/Oct/2006 – 31/Oct/2006.

Both corpora were parsed with MiniPar [3] and Syntactic Networks were built from the parsed sentences.

For the unsupervised paraphrase learning we used two seed syntactic templates for the “support” relation - “*X support Y*” and “*X praise Y*” and one seed template for the “meeting relation” - “*X meet Y*”. (We show the linear forms of the templates, in reality they are syntactic graphs as shown in Figure 1.)

Using the template learning algorithm described in section 3.1, we acquired automatically from the training corpus 6 more syntactic paraphrases for the “support” relation and 92 for the “meeting” relation.

As an example, we give the linear form of some of the “support” templates - “*X thank Y*”, “*X welcome Y*”, “*X hail Y*” and some of the “meeting” templates - “*X tell Y*”, “*X and Y hold talks*”, “*X and Y said*”, “*meeting between X and Y*”, “*discussion between X and Y*”,

“*statement by X and Y*”, “*X have dinner with Y*”, “*X give details about the talk with Y*”.

Next, using our syntactic pattern matching algorithm and both the seeds and the acquired paraphrases, we extracted from the test corpus 1'670 mentions of “support” and “meeting” events. For the purpose of the evaluation we considered only the relations between the top 33 VIP - Very Important Persons. The list of VIP was obtained by taking the people which appear most frequently in the English news in the period 01/May/2006 – 31/Oct/2006. In such a way we took an evaluation sample from 152 extracted event mentions out of the total of 1'670. Considering the evaluation schema described before, these 152 event mentions map into 40 relations between VIP, namely 33 “meeting” relations and 7 “support” relations.

In order to measure the performance of our social network extraction algorithm we identified manually all the “meeting” and “support” relations in the test corpus which hold between the 33 VIP under consideration and which are explicitly mentioned in at least one sentence. We found 75 VIP relations – 36 “meeting” and 39 “support”. In Table 1 we show the performance of our system for both type of relations when using both seeds and paraphrases.

Table 1 Performance of the Social Network extraction algorithm with paraphrases

	Precision	Recall	F1
meeting	0.606	0.556	0.580
support	0.571	0.103	0.174
overall	0.6	0.32	0.417

We measured the impact of the paraphrase learning by comparing our system's performance against a baseline relation extraction which uses only the seed templates. The results of the baseline are shown in Table 2.

Table 2 Social Network extraction with only seeds

	Precision	Recall	F1
meeting	0.818	0.25	0.383
support	0.5	0.026	0.049
overall	0.769	0.133	0.227

The overall improvement of the F1 measure when using the paraphrases is 0.19 with respect to the baseline. As expected, overall precision of the baseline is better (+0.169), since the seed templates are selected manually. It is interesting, however, that the precision for the “support” relation increases when using paraphrases.

When we add paraphrases, the overall recall increases by 0.187. For the “meeting” relation, it increases the recall twice (from 0.25 to 0.556) and for the “support” relation the effect of adding paraphrases is even more dramatic – about four times improvement (from 0.026 to 0.103).

All these numbers demonstrate that automatic paraphrase acquisition is important for relation extraction and significantly improves its performance.

Unfortunately, it was not possible to compare properly our experiments with others described in the literature, since the data and the evaluation settings were completely different from ours. However, it is worthwhile mentioning that the unsupervised approach described in [7] reports F1 in the range 0.28-0.34.

Efficiency. The whole process of matching 101 syntactic patterns against about 1'080'000 parsed sentences took about 9 minutes and 48 seconds on a PC with 2.8GHz processor Pentium 4 and 2GBytes of RAM, running Windows XP Professional SP2. It took *9 minutes and 5 seconds* to read in the memory the templates, the parsed sentences and to build the Syntactic Network. After SyntNet was built, it took only *43 seconds* to perform the rest of the pattern matching.

5.3 Using the network view

The information in the automatically extracted VIP Social Network can be used to analyze better the importance of the VIP. We run the PageRank algorithm [6] on the automatically extracted “meeting” network and found the top 5 ranked people. When using PageRank, we use names instead of pages and relations between people, instead of page references. We compared our ranking with a frequency-based method which ranks the people according to the number of articles from the test corpus in which they appear. The comparison of both methods is shown in Table 3.

Table 3 Top ranked VIP using page rank and frequency

	PageRank	Frequency ranking
1	C. Rice	G.W. Bush
2	G.W. Bush	T. Blair
3	V. Putin	C. Rice
4	E. Olmert	N. al-Maliki
5	T. Blair	S. Hussein

The importance of a person is highly subjective and it is difficult to be evaluated formally. However, intuitively it seems that PageRank works better: It returns in the top 5 a list of very important heads of states and political figures, while the frequency ranking misses some important politicians from the first list and includes “Saddam Hussein” who was much on the news titles in this period but had relatively small importance for the political world in the considered period.

6. Conclusions

We described an unsupervised syntactic approach for learning of Social Networks from news clusters. It uses two novel and efficient algorithms for syntactic template learning and matching which make our solution

potentially more scalable than other syntax-based approaches.

There is still space for improvement of our method. However, the experiments show that we can already use it for some analyst tasks.

Our approach uses syntactic templates which have well defined semantics, therefore the list of templates can be manually edited by a human expert which may improve the results reported in Table 1.

A Social Network provides an alternative view to the news topics. We used this view to measure better the importance of a person. Other interesting applications of the Social Networks include detection of groups of people and evolution of the relations inside and between the groups. Relations between people encoded in a Social Network may also be used to detect relations between news topics and navigate through news collections.

Considering these and other possible applications, we think that automatic large-scale acquisition of Social Networks will become more important in the future. In this clue, exploitation of effective and efficient algorithms for Relation Extraction is becoming an important issue.

7. References

- [1] A.-L. Barabási: *Linked: The New Science of Networks*. Perseus Publishing, Cambridge, MA, 2002.
- [2] C. Best, B. Poulighen, R. Steinberger, E. van der Goot, K. Blackler, F. Fuart, T. Oellingen, and C. Ignat: *Towards Automatic Event Tracking*, ISI, 2006.
- [3] D. Lin: *Dependency-based Evaluation of MiniPar*, Workshop on the Evaluation of Parsing Systems, 1998
- [4] Y Matsuo, J Mori, M Hamasaki, and K Ishida: *POLYPHONET: An Advanced Social Network Extraction System from the Web*, Proceedings of WWW conference, 2006
- [5] P.Mika: *Flink: Semantic Web Technologies for the Extraction and Analysis of Social Networks*, Journal of Web Semantics 2005
- [6] L.Page, S.Brin, R.Motwani, and T.Winograd: *The pagerank citation ranking: Bringing order to the Web*, Stanford publications, 1999
- [7] L. Romano, M. Kouylekov, I. Szpektor, I. Dagan, and A. Lavelli: *Investigating a Generic Paraphrase-based approach for Relation Extraction*, EAACL, Trento, Italy, 2006
- [8] N. Singh: *The Use of Syntactic Structure in Relation Extraction*, Ph.D. Thesis, 2004
- [9] I. Szpektor, H. Tanev, I. Dagan, and B. Coppola: *Scaling Web Based Acquisition of Entailment Relations*, EMNLP, 2004
- [10] H. Tanev and B. Magnini: *Weakly Supervised Approaches for Ontology Population*, EAACL, Trento, 2006
- [11] S. Wasserman and K. Faust: *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994
- [12] D. Zelenko, C. Aone, A. Richardella: *Kernel Methods for Relation Extraction*, Journal of Machine Learning Research, vol.3, 2007