

Learning Textologies: Networks of Linked Word Clusters

Hristo Tanev

Abstract Ontologies have been used in different important applications like information extraction, generation of grammars, query expansion for information retrieval etc. However, building comprehensive ontologies is a time consuming process. On the other hand, building a full-fledged ontology is not necessary for every application which requires modeling of semantic classes and relations between them. In this chapter we propose an alternative solution: learning a *textology*, that is, a graph of word clusters connected by co-occurrence relations. We used the properties of the graph for the generation of grammars and also suggest a procedure for upgrading the model into an ontology. Preliminary experiments show encouraging results.

1 Introduction

Ontologies have been used in different important applications such as information extraction, generation of grammars, query expansion for information retrieval etc. However, building comprehensive ontologies is a time consuming process. For example, the Cyc ontology encompasses about one million facts and rules manually created in a period of time of about 30 years [18]. Unfortunately, the huge amount of human effort required to build such a semantic resource is a serious disadvantage from the point of view of developers. Therefore, machine learning methods for ontology learning and population are very important, since they can reduce the time of developing an ontology. Different methods for learning ontologies were proposed recently in the literature, e.g. [12, 17]. Most of these methods require some already existing resource, such as a dictionary, which can be difficult to find, for example, for low-resourced languages and specialized domains.

H. Tanev (✉)

European Commission, Joint Research Centre, Ispra, Italy

e-mail: hristo.tanev@jrc.ec.europa.eu

On the other hand, building a full-fledged ontology is not necessary for every application which requires modeling semantic classes and relations between them.

In this chapter we propose an alternative solution which brings the ontology paradigm closer to the text. That is, we propose that semantic classes are substituted by clusters of distributionally similar words and multiwords and instead of semantic relations, we acquire co-occurrence relations, labeled with textual phrases which link the words from the related clusters. We will present several algorithms of distributional semantics as well as a link generation method, which starting from a set of seed words learn clusters of co-occurring words, represented as a *textology*: a network of word clusters, connected by co-occurrence links labeled with textual phrases.

The idea about textologies is partially inspired by the textual entailment paradigm [6], where entailment relations are derived between textual expressions rather than logical formulas or other semantic representations. In a similar way, textologies model concepts and relations on a textual level, rather than in an ontological framework.

We define a *textology* in the following way:

Definition 1 A textology is a directed labeled graph $T = \{V, E, L_v, L_e\}$ with a set of vertices V and a set of directed edges (arcs) E , connecting some pairs of vertices from V , where two or more arcs from E may link the same pairs of vertices in different directions. The functions L_v and L_e are labeling functions, defined respectively on the vertices and the arcs. L_v assigns to each vertex $v \in V$ a cluster of words and multiwords, which are lexicalizations of a concept represented by v . L_e assigns to each edge e , linking vertices v_a and v_b , a set of *connector phrases*, sequences of words which tend to connect the words from the two clusters $L_v(v_a)$ and $L_v(v_b)$. The concept lexicalizations on the vertices and the connector phrases on the links are derived from a given text corpus C .

A small textology graph from the domain of public demonstrations is shown in Fig. 1. This textology graph contains different word clusters. Let us consider some of the most important ones: Cluster *demonstrators* contains lexicalisations of the *demonstrators* concept: *protestors*, *demonstrators*, *supporters*, etc. The cluster *demands* models the concept *demand/criticize*. It contains words like *demanded*, *requested*, *criticised*, etc. Another cluster named *government* contains one word *government*. All these three clusters are linked with each other, since in the above mentioned domain of protests and demonstrations, words from these clusters tend to co-occur with each other. Let us consider the arc *demands* $\rightarrow$ *government*. It means that the words from the cluster *demands* tend to co-occur with the words from the cluster *government*, and they precede them linearly. This arc will be labeled with connectors which tend to connect *demanded*, *criticized*, *requested*, etc. with *government*. For example, the first connector is “.” which denotes immediate co-occurrence, such as in the bigram *demanded government*, *criticized government*

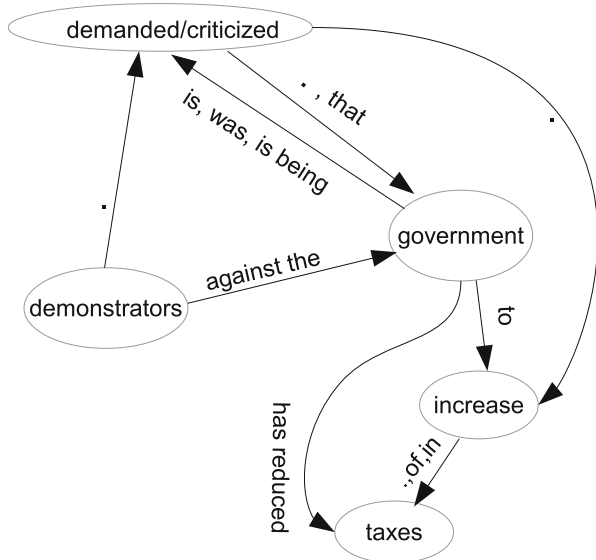


Fig. 1 Example of a textology from the domain of protests

etc., the other connector is *that*, which is used in phrases like *demanded that government*. All these collocations can be parts of phrases describing that demands or criticisms are directed towards a government. In a similar way, the opposite arc *government* $\rightarrow$ *demand* is labeled with connectors like *is, was* and *is being*, which are used in phrases like *government is criticized*, *government was criticized*, *government is being criticized*.

A textology encodes both semantic and syntactic information: the clusters represent semantic concepts and most of the paths in the textology graph encode valid phrases from the domain. This representation can be used to generate grammars for detecting domain-specific entities, situations and events. A textology can also be seen as an intermediate representation of algorithms that aim at learning an ontology. Along this line of thinking, the clusters can be considered to represent the classes of an ontology, while some of the textology arcs may be seen as a basis for the definition of semantic properties.

It is noteworthy that the representation under discussion is based on linear co-occurrences, which can be learnt from unannotated text corpora without using any language-specific processing. Therefore, our model is language-independent.

The structure of the chapter is as follows: In Sect. 2 we talk about related work. In Sect. 3 we present a method for building textologies. The method is based on several algorithms of distributional semantics. In Sect. 4 we describe a preliminary evaluation based on an experiment in which a textology is built to generate a grammar for event detection. Finally, we present our conclusion.

2 Related Work

Recently, different approaches for ontology learning and population have been proposed (for an overview see [3, 7] among others). Two approaches for learning expressive ontologies are presented in [17]: a lexical approach to generate complex class descriptions from definitional sentences and a logical approach to generate constructs of a general purpose-ontology such as disjointness axioms. A method for transforming dictionary glosses into ontology structures is presented in [12].

Concept and pattern learning are strongly related to ontology learning. However, they have been used also outside of the context of ontologies. Relevant to our work is the concept learning algorithm described in [14]. This algorithm finds concepts as sets of semantically similar words. It uses distributional clustering by applying a clustering algorithm, called CBC (*Clustering by Committees*). A good example for information extraction-related concept and pattern learning is presented in [15]. In this work, bootstrapping is introduced: as input, the system obtains a handful of lexicalizations for each concept and in every iteration it learns context patterns which are used, in turn, to obtain new concept lexicalizations. It was shown that this method succeeds in harvesting patterns and semantic classes with good levels of accuracy. Recently, a new concept and pattern learning architecture for *Never Ending Language Learning* (NELL) was developed and described in [4]. The NELL system uses the Web to learn concepts and patterns. It exploits a bootstrapping algorithm which is running continuously as a Web-based learning agent.

A common feature of the approaches mentioned so far is that they rely on language-specific parsers and part-of-speech taggers. Further, all of them work only for English.

The algorithms for building a textology are based on an underlying representation model called *word association graph*. This model was used in psychology to model human cognitive processes and human memory [5], [19]; in *Natural Language Processing* (NLP) it has been used by [2, 9, 10].

Relevant to our work is the association rule mining described in [8].

3 Building Textologies

In order to build a textology graph representation of a domain, we follow several basic steps, which include both running algorithms of distributional semantics, manual selection and cleaning of the learnt clusters. The methodology can be represented with the following schema, in which $CS_{\text{textology}}$ denotes the set of textology clusters, which in the beginning is an empty set:

1. Create an initial term set TS_{initial} by manually selecting few terms from the target domain. The term set may contain words from different parts of speech, such as nouns, proper names, verbs, adjectives, etc. There can be both single words and multiwords. As an example consider:

$$TS_{\text{initial}} = \{\text{demanded, criticized, protestors, demonstrators}\}$$

2. Create manually an initial set of semantic clusters $CS_{initial}$ from the initial term set, where a cluster represents the lexicalizations of a semantic concept, defined by the user. Each cluster contains one or more terms from the initial term set (optionally, their morphological forms can be added). In our example we will have $CS_{initial} = \{\{protestors, demonstrators\}, \{demanded, criticized\}\}$.
3. Run the automatic *cluster expansion algorithm* on the $CS_{initial}$ and expand each cluster $c \in CS_{initial}$ with additional lexicalizations of the corresponding concept. For example, our algorithm adds two more lexicalizations: *supporters* and *crowd* to the class $\{protestors, demonstrators\}$.
4. Set the list of the textology clusters $CS_{textology} \leftarrow CS_{textology} \cup CS_{initial}$.
5. Select manually a set of semantic clusters $Sel \subset CS_{textology}$ and run the *semantic context learning algorithm* which learns *context word clusters* Ctx , co-occurring with the selected cluster set Sel . For example, let us select both clusters from $CS_{initial}$, that is, in our example $Sel = CS_{initial}$. One cluster which co-occurs with the two initial clusters is $\{salary, salaries, wages\}$.
6. Each of the newly generated *context clusters* from $c \in Ctx$ is labeled with its two highest ranked words. On the basis of these two-word labels, the user preselects clusters which seem to be relevant to the target domain and application of the textology graph.
7. Each of the preselected *context clusters* is then manually reviewed and if it is relevant, it is manually cleaned and optionally the *cluster expansion algorithm* is run to expand it. In this way, a new set of relevant clusters is obtained RCS . In our example we select two new clusters, that is, $RCS = \{\{increase, drop, increases, reduction, decrease, improvement, cuts\}, \{salaries, wages, salary, wage, payments, compensation\}\}$.
8. $CS_{textology} \leftarrow CS_{textology} \cup RCS$
9. If the user thinks this is necessary, go to step 5 or step 1.
10. Run the automatic *link detection algorithm* which detects the directed edges $E_{textology}$ among the clusters in $CS_{textology}$ and the connector phrases which label them. We assume that the words in the clusters will have a similar syntactic behavior, because of the distributional similarity among them. However, this is not always true and some connector phrases will not be valid for all the words in the clusters they connect. It is up to the user to decide if she/he has to leave such connector phrases.
11. If necessary, manually edit the labels on the arcs

The final textology graph which we obtain is a graph composed of the clusters from $CS_{textology}$ and the directed edges from $E_{textology}$. The obtained graph shows how the words from the different classes tend to co-occur with each other.

This graph can be used to generate a grammar, which we show in the following sections. The grammar built from the toy example textology graph, introduced so far, can recognize phrases like *protestors demanded increase in the salaries*, *protestors criticized the cuts in the wages* etc.

The procedure for building textologies exploits two algorithms based on distributional semantics: namely the *the cluster expansion algorithm* and the *semantic context learning algorithm*. Also, a *link detection algorithm* is exploited, which finds the links between the textology clusters and their labels.

3.1 Word Association Graph

The textology concept was inspired by the notion of a word association graph, a graph model used in psychology [19] and NLP [9, 10]. In a word association graph words are related to each other on the basis of some association measure. Associations could be acquired through interviewing people or they can be gathered more indirectly by mining word co-occurrences from a text corpus, such that one can speak of *co-occurrence graphs*. This second method provides association mining on a larger scale.

In NLP, word association (or co-occurrence) graphs have been used for automatic scoring of essays [10], personal name alias extraction [1], keyword extraction [11] and studying Web communities [13]. Properties and applications of word association graphs in relation to the problem of semantic search are discussed in [2].

In general, edges in association graphs as described in the literature represent co-occurrences, which reflect generic semantic relations. Heyer et al. [9] analyzes types of semantic relations underlying these graphs.

Before learning semantic clusters and links between them, we create a word association multigraph in which two words are connected by one or more arcs, if they co-occur on a close distance. Each arc in our word association graph represents one co-occurrence pattern, which is characterized by the left and the right word (considering the linear order of co-occurrences) and the words which occur between them. In our word graph model we allow only sequences of stop words between words. We also consider a more generic co-occurrence pattern in which two words are connected if they appear on a distance of less than five tokens without considering the tokens between them.

In our approach the word associations are used as an underlying data representation, which enables the textology learning algorithms to mine for co-occurrences, which are later used for clustering, cluster expansion and learning links. It is noteworthy that we consider as vertices in word association graphs both words and multiword units, where multiwords are selected on the basis of co-occurrence patterns of their constituents. Frequencies of the occurrences of vertices and arcs are also provided by word graphs.

Note finally that textologies can be seen as extensions of word association graphs where instead of words, vertices denote clusters of words.

3.2 Algorithms

In this section, we present several algorithms for building textologies. These algorithms include the *cluster expansion algorithm*, the *semantic context learning algorithm* and the *link detection algorithm*. As pointed out earlier, these algorithms make use of a word association graph extracted from an unannotated text corpus.

3.2.1 The Cluster Expansion Algorithm

The purpose of this algorithm is to automatically expand word clusters. The algorithm takes as input a set of *seed word clusters* $C = \{c_1, c_2, c_3, c_4, \dots, c_n\}$ and expands each of them with other words and multiwords of a similar distribution profile as the corresponding seed cluster. Distributionally similar words in general represent similar concepts—they can be synonyms, for example, or have a common ancestor in their hypernym chains.

The algorithm is iterative; on each iteration three main steps are performed and the user(s) decide when the iterations are to stop. The algorithm steps are: (a) Finding contextual features. (b) Extracting new words and multiwords using the contextual features extracted in step (a). (c) Manually deleting inappropriate candidates. On each iteration we lower the threshold for selecting new words.

The *Cluster expansion algorithm* has a version called *cluster expansion in a context*, which expands the clusters from C in the context of a reference cluster cr . In practice, it is the same algorithm, but during the extraction of new words, we leave only those, which in the word association graph are adjacent to at least one word from the cluster cr . In this way, the newly learnt words for the clusters in C are related to cr . For example, if we want to learn paraphrases of the phrase *blocked the road*, we can expand a cluster with the word *blocked* in the context $cr = \{road\}$. In this way, we get additional words like *disrupted* and *marched*, which in combination with *road* mean *road blocking* as, for example, in *marched on the road* and *disrupted the road*. On the other hand, if we expand the cluster *blocked* without context, we get additional words like *stop* and *prevent* which may be related to *blocked*; now, they cannot be used to express a road blocking situation.

It is noteworthy that the selection threshold for this version of the algorithm is lower than in the case of the non-contextualized expansion. This is because we have an additional criterion, that is, *reference class*, which ensures a better accuracy.

Learning Contextual Features

For each semantic class c_i we iterate through its members words and multiwords, denoted by $lexicalitems(c_i)$. For each lexical item $lxi \in lexicalitems(c_i)$ we extract *contextual features* from each arc and node adjacent to lxi in the word association graph. The contextual features are two types: left and right features.

Left features are formed from the arcs which go into lxi ; they are formed from the adjacent words or multiwords, concatenated with the connector sequence on the arc which connects them to lxi . Left features are n -grams which appear immediately on the left from lxi in the text corpus from which the word graph is created. In a similar way we extract from the word association graph the so-called right features which appear on the right side of lxi .

Each contextual feature of a class c_i is assigned a score which shows how well it co-occurs with the seed terms. The score is calculated as a sum of the co-occurrence score of the feature with respect to each lexical item from c_i . The co-occurrence with the single lexical items is based on the pointwise mutual information. The details of feature scoring are based on a formula described in [16]. As an example, some of the top-scoring left contextual features of the class *protestors* are *disperse the*, *tear gas at*, *force against* and some of the top-scoring right context features are *chanted*, *carried signs*, *had gathered*.

Next, we take for each cluster $c_i \in C$ the top scoring features and merge them into a contextual-feature pool, which constitutes a semantic space where the clusters are represented. The contextual features for each cluster c_i form a *context vector* $v_{context}(c_i)$ which represents the semantics of c_i through its typical contexts. Each dimension in this vector is a contextual feature, extracted for any of the considered clusters and the co-ordinates are the scores calculated with our algorithm.

Learning New Lexical Items

After contextual features are learnt for each seed cluster from C , our procedure extracts new lexical items which tend to co-occur with these contextual features. In this way, we obtain lexical items which have a similar distributional profile as the seed clusters. We use the word association graph to extract the nodes which co-occur with each contextual feature extracted on the previous step.

Our algorithm represents each lexical item lxi as a vector $v_{context}(lxi)$ in the space of contextual features. The dimensions of this semantic space are defined by the set of contextual features extracted before. The coordinate of a lexical item lxi with respect to the dimension f reflects the co-occurrence trend between lxi and the contextual feature f . For more details of this procedure see [16].

Our term learning approach calculates the relevance of a lexical item lxi for a category c , using the following formula

$$termscore(lxi, c) = \frac{v_{context}(lxi) \cdot v_{context}(c)}{|v_{context}(c)|} \quad (1)$$

This relevance value computes the projection of the lexical item vector on the category vector. Candidates which are above a certain threshold are returned to the user.

3.2.2 Semantic Context Learning

The goal of this algorithm is to find clusters of words and multiwords which tend to co-occur with the lexical items from one or more clusters. This algorithm allows us to expand the textology graph beginning from few clusters. The algorithm accepts as input a set of clusters $C = \{c_1, c_2, \dots, c_n\}$ and a number k . It learns clusters of words which tend to co-occur with at least k input clusters. The semantic context learning works in two main steps: (a) Learn words and multiwords co-occurring with at least k clusters from C and (b) cluster these lexical items using their contextual features.

Learning Words and Multiwords Co-occurring with the Input Clusters

In this step the algorithm extracts from the word graph all those words that co-occur with the input clusters. The co-occurrence score of each lexical item for each cluster is calculated by analogy to the feature scores within the cluster expansion algorithm. The final score of each lexical item is the sum of its co-occurrence scores with respect to the different clusters under consideration. Lexical items that are co-occurring with less than k clusters are discarded.

Finding Clusters

In this step, the algorithm first finds the vectors of contextual features of the lexical items extracted in the previous step. Next, the lexical items are clustered using the cosine similarity between these vectors. The feature vector is computed the same way as in the cluster expansion algorithm. We perform agglomerative clustering based on bottom-up average-linkage. Finally, the clusters are ordered by taking into account the highest scoring lexical item in each cluster. For example, the top-scoring context clusters for $C = \{\{protest\}, \{school, schooling, education\}\}$ and $k = 2$ are $\{children, kids\}$, $\{parents, mothers, firefighters\}$ and $\{teachers, faculty, volunteers, professors\}$, while the top scoring context clusters for $C = \{\{protestors\}\}$ and $k = 1$ are $\{police, gardai\}$, $\{streets\}$ and $\{disperse, quell\}$.

3.2.3 Link Detection Algorithm

This algorithm detects arcs between pairs of clusters of the textology. To this end, the algorithm explores the word association graph for arcs between lexical items that belong to the textology clusters. More precisely, the algorithm takes as input a set of clusters of a textology, which are already learnt: $C = \{c_1, c_2, \dots, c_n\}$. Then, for each pair of clusters c_i and c_j it finds all arcs from the word association graph that connect each pair of lexical items $w_a \in c_i$ and $w_b \in c_j$. The arcs, which are labeled by the same connector words and which link words from the same pair of clusters in

the same direction, are merged to what we call a *generalized arc*. More precisely, we create a generalized arc from cluster c_i to cluster c_j , labeled by a connector phrase l , if there is more than one arc in the word association graph, labeled with l and linking two different pairs of lexical items (w_a, w_b) , such that $w_a \in c_i$ and $w_b \in c_j$. Here, we put a restriction for more than one arc based on empirical observations. However, depending on the corpus or the application a higher threshold could be applied and more equally labeled arcs may be required in order to produce a generalized arc. The generalized arcs become arcs of the textology. Finally, the user can manually delete some of the learnt arcs if necessary. As an example, consider $C = \{c_1, c_2\}$, where $c_1 = \{\text{protest}\}$ and $c_2 = \{\text{school, schools, education}\}$. In this case, the arcs generated by our algorithm are

$$\{\text{school, schools, education}\} \xrightarrow{\text{in}} \{\text{protest}\} \quad (2)$$

$$\{\text{protest}\} \xrightarrow{\text{against}} \{\text{school, schools, education}\} \quad (3)$$

4 Using Textologies

4.1 From Textologies to Ontologies

The classes in a textology represent concepts which in the context of ontologies can be represented as classes. For example, in Fig. 1, from the textology we can generate the concepts *demand/criticize*, *government*, *demonstrators*, *increase* and *taxes*. Then, more general classes can be introduced together with the corresponding *is-a* relations between them. For example, we can create the class *people* as a hypernym of *demonstrators*, *institution* as a hypernym of *government* and *action* as a hypernym of *increase* and *demand/criticize*.

Occasionally, several clusters can be used to create one complex semantic class. For example, the clusters *government*, *increase* and *taxes* and the links between them describe phrases which may motivate the introduction of a concept named *institutional-action*. Moreover, some of the clusters, which represent binary predicates may be modeled as properties, if they express fundamental relations between concepts. This, however, depends on the ontology.

Then, some of the relations among clusters can be used as a basis for creating properties. Note that arcs in a textology represent syntagmatic relations, rather than semantic ones. Thus, some of them will not be transformed into properties. Further, many arcs may be merged into a single property. Looking at the aforementioned example, we can define the property *agent-of-demand/criticism* of the domain *demand/criticise* and range *people*. Also the property *target-of-demand/criticism* may be generated which has the same domain as the previous property while its range is *institution*.

It is noteworthy that preposition connectors usually represent different relations, however they are limited in the range of the introduced semantic relations. For example, the preposition *of* may introduce a property, e.g. *the price of the smartphone*, a *part-of* relation, e.g. *the display of the smartphone*, or ownership, e.g. *the car of Mr. Orlov*. In order to deduce the semantic relation from a surface co-occurrence like *X of Y*, one can consider the semantics of the words which fill the *X*- and the *Y*-slot of this pattern. For example, words like *length* and *price* as instances of the *X*-slot designate properties, while in cases where nouns that refer to objects occur in the *X*-slot, the pattern refers to instantiations of the *part-of*-relation, e.g. *the roof of my home* or ownership *the office of the boss*—depending on the word which fills the *Y* slot. If it is a non-animate object like *home* that occurs in the *X*-slot, then we have most probably a *part-of*-relation, while in the case of nouns denoting animate objects like *boss*, we most probably get an instance of the *possession*-relation.

While the method described so far is quite general, we think that creating a textology can be an important step in building an ontology.

4.2 Grammar Generation

Grammars provide a natural mechanism to parse text and link it to semantic concepts and properties. We present in this section a grammar learning algorithm, which is based on the textology model. For us a grammar is a set of production rules of two types: (1) Rules of the type $C \rightarrow (w_1, w_2, \dots, w_n)$, where C designates a cluster and $w_1 - w_n$ are words from this cluster. (2) Rules of the type $A \rightarrow C_1 \text{connector}_1 C_2 \dots$ which encode more complex expressions. There C_i is a symbol, referring to a cluster and connector_i refers to a connector phrase. The grammars we learn can be considered to be lexicalized grammars. They do not encode the linguistic structure of the expressions and do not make reference to parts of speech or syntactic categories. Rather than that, they refer to the clusters from the textology and the lexicalizations from these clusters.

There are two important steps in creating grammars from a textology: First, each cluster can be represented as a non terminal symbol which generates the words which are elements of the cluster. Second, some arcs or paths in the textology can be transformed into context-free grammar rules representing more complex phrases. For example, in Fig. 1 the clusters *demand/criticise* and *demonstrators* can give birth to the following production rules, which generate terminal strings:

- (1) *DemandCriticise* $\rightarrow$ (*demanded*|*criticised*|*demanding*|*seeking*|...)
- (2) *Demonstrators* $\rightarrow$ (*demonstrators*|*protestors*|...)

Then, the textology arc *demonstrators* $\rightarrow$ *demand/criticise* may give birth to the higher level rule (3), which parses more complex phrases like *demonstrators demanded*.

- (3) *DemonstratorsDemand* $\rightarrow$ *Demonstrators DemandCriticise*

We use the following algorithm to generate grammar rules:

1. Manually select the set of paths P from which rules are to be generated. Suppose, we select only one path: *demonstrators* $\rightarrow$ *demanded/criticised* $\rightarrow$ *government*
2. All the clusters on any paths from P are stored in the set C . In our example, $C = \{\textit{demonstrators}, \textit{demanded/criticise}, \textit{government}\}$.
3. For each cluster $c \in C$, create a rule, which generates as terminal strings all the words from the cluster. For example, *Demonstrators* $\rightarrow$ (*demonstrators|protestors|...*).
4. For each $p \in P$, generate a grammar rule, where on the righthand side each cluster from P is represented as a non-terminal symbol, the non-terminals are ordered as the corresponding clusters appear in p and if between two clusters the arc is tagged with a symbol, different from “.” (which means immediate co-occurrence), then the labels are inserted as terminal strings between the corresponding non terminals. In our example, the following rule will be generated: $A \rightarrow \textit{Demonstrators DemandedCriticised Government}$. In this rule there are no terminal strings on the righthand side, since all the arcs on the path are tagged with “.”. The symbol A on the left was chosen arbitrarily.

Following the described procedure, one can easily generate one or more grammars from a textology by selecting different paths.

Grammars generated with this procedure have some limitations: They cannot contain recursion and also they work on only two levels—the first level generates terminal strings and the second level generates more complex phrases from these terminal strings. Although the complexity of the grammars generated from the aforementioned procedure is limited, we show here that these grammars can be used for practical purposes.

5 Experiments and Evaluation

5.1 Building a Textology

In order to test the feasibility of our method, we built a textology which represents domain knowledge about protests and civil disorders. In this experiment we concentrated in particular on the violence and demands of protestors. In order to build this textology we indexed as a word association graph about 100,000 news articles. The most important participants in this type of events are the protestors, therefore we used as an initial term set for the textology learning algorithm the set $\{\textit{protestors}, \textit{demonstrators}\}$.

We run the cluster expansion algorithm on this seed set with five iterations, the system suggested six new terms altogether. Five of these terms were correct, i.e. could be considered lexicalisations of the concept “protestors”. In this clue, the accuracy for the expansion of this initial seed class was found to be **83 %**.

Table 1 Accuracy of cluster expansion

Algorithm	Average number correct learnt words	Average accuracy
Cluster expansion	1.9	0.74
Cluster expansion in context	2.72	0.33

Using the expanded clusters *protestors*, we run the context learning algorithm which found about 300 clusters of words which tend to co-occur with the initial seed class. We looked through the top 100 clusters, where we looked at the top two words from each cluster. In this way we chose 25 clusters for further review. From these, we found 20 clusters to be very relevant to the domain of violence in protests and civil disorders. Some of the remaining clusters had some relation with the target domain, however they were not as related as the 20 selected ones. Then, we run the cluster expansion for these 20 clusters. The accuracy of the expansion is shown in the first row of Table 1. We did not evaluate the expansion for the other 280 clusters, since the idea of our approach is that the user selects and works with only a subset of the clusters.

In order to include in our textology clusters related to protestors' demands, we performed a second learning iteration: In this case, the initial term class was set to be *{demanded, criticized}*. We run two iterations of the cluster expansion, this time it was done in the context of the class *protestors*. In this way, this algorithm learnt 16 new terms. It turned out ten of these were correct, thus the accuracy of the algorithm turned out to be **63 %**.

We run the context learning algorithm, this time using two clusters *demanded* and *protestors*. The output of our algorithm were clusters, co-occurring both with the words from *protestors* and *demands*. While many clusters were relevant to the topic *protestors' demands*, we chose 12 which were the most common reasons for demands and criticisms of the protestors. These clusters were *{deal}*, *{money, funds}*, *{law, bill}*, *{school}*, *{prosecution}*, *{media}*, *{health}*, *{job}*, *{cut}*, *{tax}*, *{pay, paid, paying}*, *{bank}*. We also performed an additional run with the clusters *demanded* and *protest*. We added one additional cluster to our textology, namely *wage, salary, salaries, wages, allowance, allowances*.

The clusters were expanded in one iteration, using the cluster expansion algorithm in the context of the *protestors* cluster. The accuracy of the expansion is shown in the second row of Table 1. As we mentioned earlier, the threshold of the expansion in context is lower than the expansion without context, due to the presence of a reference cluster as a constraint. Because of these differences, the accuracies of both versions are not comparable.

Finally, we ran the link extraction algorithm which found arcs connecting clusters generated so far. The obtained textology was used for generation of several grammars in the domain of protests and civil disorders.

5.2 Generating Grammars

We developed three grammars using the grammar generation algorithm described in Sect. 4.2. These grammars aimed at extracting violence-related events, road blocking and expressions of economic demands during a protest. In case of these three grammars, we experimented with one-edge paths.

Regarding the grammar about violence related to protests, we selected the arcs between 35 pairs of clusters from the set $\{\{disperse, quell\}, \{police, gardai\}, \{clashes, fighting, \dots\}, \{threw, hurled, \dots\}, \{tear\ gas, rubber\ bullets, \dots\}, \{fired, shooting\}, \{clash, clashed\}, \{injured, hurt, \dots\}, \{stones, grenades, \dots\}, \{rallies, demonstrations, \dots\}, \{protesting, protest\}, \{forces, troops\}\}$. Regarding the grammars about road blocking, we selected the arcs between the clusters $\{blocked, blocking\}$ and $\{road, roads, highway, highways\}$. Finally, in case of the grammar about economic demands we used the arcs between 11 pairs from the set of clusters: $\{\{protesting, protest\}, \{demanded\}, \{protestors\}, \{job\}, \{cuts\}, \{tax\}, \{pay, paid, paying\}, \{jobs, employment\}, \{salaries, allowances\}, \{rallies, demonstrations\}, \{wage, salary, \dots\}\}$.

For each grammar, we built a small test set which contained both *positive* sentences which express the events of interest and *negative* sentences, which though being selected from the domain of protests, do not express events of interest. The corpus concerning protest-related violence contained 24 positive and 11 negative sentences; the corpus about road blocking contained 21 positive and 19 negative sentences; the corpus about economic demands contained 15 positive and 15 negative sentences.

We run each grammar on the corresponding test set and considered that a sentence is selected when at least one grammar rule matches a substring of it. In this way, we evaluated the task for selecting sentences describing events. We measured the precision, recall and F-measure for each grammar. The results are shown in Table 2. Precision is 100 % due to the unambiguous nature of the generated rules. Obviously, recall can be improved.

Table 2 Accuracy of detecting event sentences

Event type	Precision (%)	Recall (%)	F-measure (%)
Violence	100	46	63
Road blocking	100	33	50
Economic demands	100	40	57

6 Conclusion

We presented a knowledge representation model called *textology*. The model is less expressive than formal ontologies, but nevertheless maps semantic classes and their syntagmatic relations. Therefore, textologies provide an expressive format of an intermediate level which though being less expressive than ontologies, allow for tasks like grammar generation. Textologies may also be considered for upgrading them into full-fledged ontologies. We presented several algorithms for building textologies and generating context-free grammars based thereon. Our preliminary experiments show encouraging results.

In future work, we aim at experiments with other algorithms of knowledge acquisition and grammar learning. Additionally, we aim at NLP applications based on our model.

References

1. Bollegala D, Matsuo Y, Ishizuka M (2008) A co-occurrence graph-based approach for personal name alias extraction from anchor texts. In: International joint conference on natural language processing (IJCNLP), pp 865–870
2. Bordag S, Heyer G, Quasthoff U (2003) Small worlds of concepts and other principles of semantic search. In: Böhme T, Heyer G, Unger H (eds) *Iics*, vol 2877. Springer, New York, pp 10–19. Retrieved from <http://dblp.uni-trier.de/db/conf/iics/iics2003.html#BordagHQ03>
3. Buitelaar P, Cimiano P (eds) (2008) *Ontology learning and population. Bridging the gap between text and knowledge*. Springer, Berlin
4. Carlson A, Betteridge J, Kisiel B, Settles B, Estevam R, Hruschka J, Mitchell T (2010) Toward an architecture for never-ending language learning. In: Proceedings of the twenty-fourth AAAI conference on Artificial Intelligence (AAAI-10), Atlanta, GA, pp 1306–1313
5. Costa ME, Bonomo F, Sigman M (2009) Scale-invariant transition probabilities in free word association trajectories. *Front Integr Neurosci* 3:17
6. Dagan I, Glickman O, Magnini B (2006) The pascal recognising textual entailment challenge. In: *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising textual entailment*. Springer, New York, pp 177–190
7. Drumond L, Girardi G (2008) A survey of ontology learning procedures. In: *The 3rd workshop on ontologies and their applications*, Salvador, Brasil, pp 13–25
8. Feldman R, Sanger J (2007) *The text mining handbook. Advanced approaches in analyzing unstructured data*. Cambridge University Press, Cambridge
9. Heyer G, Läuter M, Quasthoff U, Wittig T, Wolff C (2001) Learning relations using collocations. In: Maedche A, Staab S, Nedellec C, Hovy EH (eds) *Workshop on ontology learning*, vol 38. CEUR-WS.org. Retrieved from <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2001ol.html#HeyerLQWW01>
10. Klebanov BB, Flor M (2013, August). Word association profiles and their use for automated scoring of essays. In: *Proceedings of the annual meeting of the association for computational linguistics*, Sofia, Bulgaria
11. Matsuo Y, Ishizuka M (2004) Keyword extraction from a single document using word co-occurrence statistical information. *Int J Artif Intell Tools* 13(01):157–169
12. Navigli R, Velardi P (2008) From glossaries to ontologies: extracting semantic structure from textual definitions. In: *Ontology learning and population. Bridging the gap between text and knowledge*. Springer, Berlin, pp 71–87

13. Ohsawa Y, Soma H, Matsuo Y, Matsumura N, Usui M (2002) Featuring web communities based on word co-occurrence structure of communications: 736. In: Proceedings of the 11th international conference on world wide web, p 742
14. Pantel P, Lin D (2002) Discovering word senses from text. In: Proceedings of ACM SIGKDD conference on knowledge discovery and data mining, Edmonton, pp 613–619
15. Riloff E, Jones R (2002) Learning dictionaries for information extraction by multi-level bootstrapping. In: Proceedings of the sixteenth national conference on Artificial Intelligence (AAAI 99), Orlando, FL, pp 474–479
16. Tanev H, Zavarella V, Kabadjov M, Piskorski J, Atkinson M, Steinberger R (2009) Exploiting machine learning techniques to build an event extraction system for Portuguese and Spanish. *Linguamatica* 2:55–66
17. Völker J, Haase, P Hitzler P (2008) Learning expressive ontologies. In: Proceedings of the 2008 conference on ontology learning and population: bridging the gap between text and knowledge. IOS Press, Amsterdam, pp 45–69
18. Wikipedia: Cyc. (2014) Retrieved from <http://en.wikipedia.org/wiki/Cyc>
19. Zortea M, Menegola B, Villavicencio A, Salles JFD (2014) Graph analysis of semantic word association among children, adults, and the elderly. *Psicologia: Reflexão e Crítica* 27(1):90–99

Text Mining

From Ontology Learning to Automated Text Processing
Applications

Biemann, C.; Mehler, A. (Eds.)

2014, X, 238 p. 50 illus., 23 illus. in color., Hardcover

ISBN: 978-3-319-12654-8