Inducing Classes of Terms from Text

Pablo Gamallo [*], Gabriel P. Lopes, and Alexandre Agustini

[1] Faculdade de Filologia, Universidade de Santiago de Compostela, Spain
pablogam@usc.es
[2] Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portgual
gpl@di.fct.unl.pt
[3] Departamento de Informática, PUCRS, Brazil
agustini@inf.pucrs.br

Abstract. This paper describes a clustering method for organizing in semantic classes a list of terms. The experiments were made using a *POS* annotated corpus, the ACL Anthology, which consists of technical articles in the field of Computational Linguistics. The method, mainly based on some assumptions of *Formal Concept Analysis*, consists in building bi-dimensional clusters of both terms and their lexico-syntactic contexts. Each generated cluster is defined as a semantic class with a set of terms describing the extension of the class and a set of contexts perceived as the intensional attributes (or properties) valid for all the terms in the extension. The clustering process relies on two restrictive operations: *abstraction* and *specification*. The result is a concept lattice that describes a domain-specific ontology of terms.

1 Introduction

This paper describes a method for clustering terms into semantic classes using as input a domain-specific corpus and a preliminary list of terms in the same domain. The corpus is the ACL Anthology, which consists of technical articles published by journals and conferences on Computational Linguistics. The method consists in building bi-dimensional clusters of both terms and their properties. Each cluster is the result of either merging or unified their constituents (i.e., terms and properties). The properties of a cluster/class are represented by those lexico-syntactic contexts co-occurring in the corpus with all terms of the class.

The basic intuition underlying our approach is that similar classes of terms can be aggregated to generate either more specific or more generic classes, without inducing odd associations between terms and their properties/contexts. A new *specific* class is generated when the properties of the constituent classes are merged (intension expansion), while the terms are intersected (intension reduction). A new *generic* class is generated when the properties are intersected (intension reduction), while the constituent terms are merged (extension expansion). Intersecting either terms or properties allows us to generate tight clusters with representative and prototypical constituents. These

[*] This work has been supported by the Spanish Government, within the project GaricoTer.

tight clusters can be perceived as centroids to classify both new terms and properties. The theoretical background our work is mainly based on is *Formal Concept Analysis* (FCA) [5, 9]. The clusters we acquired have all the features of "formal concepts" in FCA. Figure 1 shows a class consisting of a set of terms and a set of properties learnt by our system. The cluster represents a formal concept with a term extension ("Natural Language Processing", "Speech Processing", etc) and a descriptive intension ("research in", "area of", etc). The clustering algorithm only selects those properties that can co-occur with all terms in the extensional set. Each crossing line in the figure represents the binary relation "co-occurs with" between a property (or local context) and a term of the class.

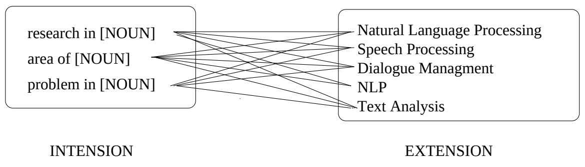

Fig. 1. An example of bi-dimensional cluster generated by our method

Let's note that our algorithm learnt the main areas in NLP are Text, Speech, and Dialogue. This is in accordance with the *TSD* conference name.

This article is organised as follows. Section 2 introduces some related work. Then sections 3 and 4 describes two complementary clustering methods: by abstraction and by specification. Finally, in section 5, experiments, results, and an evaluation protocol are discussed.

2 Related Work

In order to induce semantically homogeneous clusters of words (tokens, types or lemmas), some approaches compare the semantic similarity between $< word, context >$ pairs and sets of those pairs. These sets are perceived as semantic classes, also called "selection types" [8, 11]. Given two vocabularies, W and LC, which represent respectively the set of words and the set of local contexts, a semantic class is defined as a pair $< LC', W' >$, where $LC' \subseteq LC$ and $W' \subseteq W$. In this model, the same word or context can in principle belong to more than one class. So, the positive side of these approaches is that they try to take into account polysemy. Some difficulties arise, however, in the process of class generation. Those approaches propose a clustering algorithm in which each class is represented by the centroid distributions of all of its members. This is in conflict with the fact that many words and local contexts can significatively involve more than one semantic dimension. As a result, the clustering method appears to be too greedy since it overgenerates many wrong associations between words and local contexts.

To avoid this problem, a more recent approach tried to limit the information contained in the centroids by introducing a process of "clustering by committee" [7]. The centroid of a cluster is constructed by taking into account only a subset of the cluster members. This subset, called “committee”, contains the more representative members (prototypes) of a class. So, the main and more difficult task of such an approach is to first identify a list of committees, i.e., a list of semantically homogeneous clusters. Committees represent basic semantic classes of similar words and are useful for word classification.

Other approaches also try to identify homogenous clusters representing basic semantic classes. The main difference with regard to the former methods is that each basic cluster is constituted, not by similar words, but by a set of similar local contexts [3, 1, 10, 4]. The method is focused on computing the semantic similarity between local contexts. Words are no more seen as objects to be clustered but as features of contexts. These are taken as the objects of the clustering process. As local contexts turn out to be less polysemic than words, it is assumed that searching for classes of homogeneous contexts is an easier task than to find tight classes of semantically related words. The main drawback, however, is that local contexts are less frequent than words and, then, they are more sparse.

The method proposed in this paper considers that clusters are bi-dimensional objects consisting of both words (or terms) and contexts (or properties). Our main contribution is the use of very restrictive operations (*specification* and *abstraction*) in the process of building tight clusters. Using these operations, we aim at solving the overgeneration problem. In the next section, we will describe a clustering algorithm that makes use of the abstraction operation.

3 Clustering by Abstraction

It consists in building generic classes from very specific clusters.

3.1 The Input: Specific Classes

The input of this clustering process is a list of very *specific* classes of terms, i.e., each class consists of a small set of terms (small extension) and a large set of properties/contexts (big intension). To build these specific classes, three tasks are required: first, the training corpus is annotated to extract co-occurrences between word (or multi-word) lemmas and their lexico-syntactic contexts. Second, a list of terms which are relevant in the domain is selected. And third, we compute similarity between the terms of this list and all word (or multi-word) lemmas occurring in the corpus.

Corpus Processing The corpus is first *POS* tagged and then binary dependencies are extracted using pattern matching techniques (articles and pronouns are previously removed). From each binary dependency, two complementary lexico-syntactic contexts are selected. For instance, given a binary dependency: “entry-in-thesaurus”, two contexts/properties are selected: $<$ entry in [NOUN] $>$ and $<$ [NOUN] in thesaurus $>$. We follow the notion of *co-requirement* introduced in [4].

Finally, each lexico-syntactic context is associated to its co-occurring word lemmas to build a collocation database. Each word lemma can be viewed as a vector and each lexico-syntactic context correspond to a feature. Before starting the clustering process, sparse contexts are removed from the vectors. A context is sparse if it has high word dispersion. Dispersion is defined as the number of different word lemmas occurring with a lexico-syntactic context divided by the total number of different word lemmas in the training corpus. So, the vector space is only constituted by those lexico-syntactic contexts whose word dispersion is lower than an empirically set threshold.

List of Terms The startpoint to build the specific classes is a list of terms. This list can be selected by manual intervention using pre-existing glossaries, or by unsupervised analysis of domain-specific documents (terminology extraction). In our experiments, we used as startpoint an existing glossary of terms.

Similarity To build the input classes, we compute the Dice similarity between each term and the rest of word and multi-word lemmas. Similarity between a term, t, and a word or multi-word lemma, w, which is not in the starting list of term samples, is computed as follows:

$$Dice(t, w) = \frac{2 * \sum_i min(f(t, c_i), f(w, c_i))}{f(t) + f(w)}$$

where $f(t, c_i)$ represents the number of times t co-occurs with the context c_i. Likewise, $f(w, c_i)$ represents the number of times w co-occurs with the context c_i. For each term, its top-*k* most similar lemmas, where $k = 5$, are selected. For instance, in our experiments on a corpus consisting of articles published in conferences and journals on Computational Linguistics, the 5 most similar words to the term “thesaurus” are “bilingual lexicon”, “bilingual dictionary”, “taxonomy”, “lexical resource”, and “ontology”. As it was expected, words that are similar to a given term in a specific domain are also terms in that domain.

Now, a set of specific classes can be generated. Given the top-5 most similar lemmas to a given term, we build 5 very specific classes by aggregating the term to each similar word in the list. The intension of each class consists of those contexts that are shared by the two similar terms. Table 1 shows the five basic classes associated to “thesaurus”.

The specific classes built from the previously selected terms are the input of the clustering process.

3.2 Generating Intermediate Classes

The first step of the clustering process is to build a set of “intermediate classes” (neither very specific nor very abstract). For this purpose, we use a clustering algorithm inspired by [2]. To explain this algorithm, let’s take the specific classes in Table 1. The first basic class, 0110, is considered as the starting centroid. Then, we search for other centroids among the other 4 classes. A class is a centroid if it is not similar to the previously identified centroids. Here, we consider that two objects are similar if they share more than 50% of the properties/contexts. In our example, there is only one centroid more: class 0112. Finally, each one of the remaining classes is aggregated to a centroid if

Table 1. 5 specific classes built from the term "thesaurus"

class	extension	intension
0110	{thesaurus, bilingual_lexicon}	{<access to [N]>, <[N] construction>, <entry in [N]>, <machine-readable [N]>, <compilation of [N]>, <headword of [N]>, <online [N]>, <consult [N]>, ... }
0111	{thesaurus, bilingual_dictionary}	{<access to [N]>, <term in [N]>, <entry in [N]>, <machine-readable [N]>, <compilation of [N]>, <[N] entry>, <headword of [N]>, <online [N]>, <consult [N]>, <define in [N]>, ...}
0112	{thesaurus, taxonomy}	{<[N] relation>, <[N] construction>, <concept of [N]>, <relation in [N]>, <node of [N]>, <node in [N]>, <[N] generation>, <[N] concept>, <[N] from dictionary>, <monolingual [N]>, <[N] of domain>, ...}
0113	{thesaurus, lexical_resource}	{<access to [N]>, <word from [N]>, <entry in [N]>, <machine-readable [N]>, <online [N]>, <consult [N]>, <define in [N]>, <computerized [N]> ...}
0114	{thesaurus, ontology}	{<[N] relation>, <[N] construction>, <concept of [N]>, <relation in [N]>, <node of [N]>, <class in [N]>, <[N] generation>, <[N] concept>, <[N] from dictionary>, <monolingual [N]>, <[N] of domain>, ...}

they are similar, i.e. if thy share more than 50% of their contexts. In our example, both 0111 and 0113 are aggregated to 0110, while 0114 is joined to 0112. All aggregations are made using the operator of abstraction since each generated cluster is obtained by intersecting the two constituent intensions. Following this algorithm, we obtain two intermediate classes (see Table 2)

Table 2. Intermediate classes built from "thesaurus"

class	extension	intension
CL_015	{thesaurus, bilingual lexicon, bilingual dictionary, lexical resource}	{<access to [N]>, <[N] construction>, <entry in [N]>, <machine-readable [N]>, <headword of [N]>, <online [N]>, <consult [N]>, ... }
CL_123	{thesaurus, taxonomy, ontology}	{<[N] relation>, <[N] construction>, <concept of [N]>, <relation in [N]>, <node of [N]>, <[N] generation>, <[N] concept>, <[N] from dictionary>, <[N] of domain>, ...}

Let's note that this clustering algorithm allows us to put the same term in different classes (soft clustering). Terms, even if they are well-defined technical expressions, can be used in a corpus with a high degree of ambiguity. For instance, "thesaurus" is considered in our training corpus either as a repository of entries (lexical resource) or as a concept structure (ontology). Our algorithm found other polysemous terms. For instance, “thematic role” was aggregated with “case slot” into a class characterised by the property <fill [N]>, whereas it was put into another cluster with terms such as “grammatical function” or “semantic relation”, where they share the property <assignment of [N]>. In sum, a thematic role can be viewed either as a recipient (slot) to be filled by an entity or as a linguistic function the entity is assigned to.

Furthermore, the generated clusters contain word lemmas that were not in the starting list of terms (e.g., “hierarchy” and “bilingual lexicon”). Indeed, lemmas appearing with terms in a cluster must also be considered as terms. This clustering process is repeated for the remaining input classes associated to the other terms of the list. Intermediate classes are the input of the following clustering step.

3.3 Generating Generic Classes by Hierarchical Clustering

A standard hierarchical clustering takes as input the intermediate classes to generate more generic ones. For this purpose, we make use of an open source software: Cluster 3.0[1]. In this step, the clustering conditions are the same: the similarity threshold is still 50%, and classes are aggregated with the operation of abstraction. Table 3 illustrates two generic classes containing “thesaurus” as a member. They are the result of putting together the intermediate classes depicted above in Table 2 with other similar intermediate classes.

Table 3. Generic classes built from “thesaurus”

class	**extension**	**intension**
$NODE\_09$	{thesaurus, automatic thesaurus, bilingual terminology, terminology, bilingual lexicon, bilingual dictionary, lexical resource}	{<[N] construction>, <entry in [N]> }
$NODE\_21$	{thesaurus, hierarchical structure, hierarchy, taxonomy, ontology, type hierarchy}	{<concept of [N]>, <node of [N]>, <[N] of domain>}

3.4 Classifying Unknown Words

So far, the generated clusters have been loosing relevant information step by step, since they were aggregated using intersecting operations. Besides that, the intersecting aggregations did not allow us to infer context-word associations that were not attested in the training corpus. As has been mentioned above, our objective was to design a very restrictive clustering strategy so as to avoid overgeneralisations.

In order to both reintroduce lost information and learn new context-word associations, the last step aims at assigning unknown words to the generic classes generated by the clustering algorithm. An unknown word is assigned to one or more classes if it

[1] *http://bonsai.ims.u-tokyo.ac.jp/˜mdehoon/software/cluster/software.htm*

satisfies two conditions: 1) it co-occurs with more than 50% of the contexts constituting the class intension, and 2) it is one of the top-10 most similar words to, at least, one of the terms in the class extension. We assume that those words being correctly classified should been considered as terms. So, classification can also be perceived as a kind of automatic term extraction.

4 Clustering by Specification

The only difference with regard to the former strategy is that the collocation database is viewed now as a collection of word vectors. Each unique context corresponds to a vector and each word lemma corresponds to a feature. The input classes are then generic concepts whose intension consists of two similar context/properties (instead of two similar terms). The extension is the set of terms and word lemmas shared by those contexts. For instance, <entry in [N]> was considered to be the most similar context to <[N] entry>. Both contexts represent the intension of a generic class whose extension consists of terms such as “bilingual dictionary”, “bilingual lexicon”, etc. Further clustering steps will make this generic class more specific (by intersecting the extension and unifying the intension with other similar classes). As a result, a very specific class is generated. The main problem of this algorithm is that lexico-syntactic contexts are more sparse than word lemmas. Classification of unknown contexts was not performed. This task is part of our current research.

5 Experiments and Evaluation

Experiments have been carried out over a large corpus of 25 million word tokens: the ACL Anthology[2]. It consists of technical articles published in conferences and journals in the domain of Computational Linguistics. The main drawback is that the corpus is very noisy because of the .pdf to .txt conversion process. *POS* tagging was made with freely available Tree-Tagger[3].

The starting glossary of terms contains 175 entries. It is a manual selection from the glossary appearing in the appendix of the “Oxford HandBook of Computational Linguistics”, edited by Ruslan Mitkov[4]. We learnt 201 generic classes using the abstraction algorithm. These classes contain 803 different terms. So, as well as being organised terms in classes, we extracted more than 600 new terms. On the other hand, the specification algorithm gave rise to 803 specific classes with 297 different terms.

Measuring the correctness of the acquired classes of terms is not an easy task. We are not provided with a gold standard against to which results can be compared. So, we evaluated the capacity of the algorithm to classify unknown terms into existing generic clusters and the quality of the clusters themselves. The test data consisted of 160 randomly selected classifications. Then they were given to 3 human judges for evaluation.

[2] *http://wing.comp.nus.edu.sg/ min/dAnth/acl/*

[3] *http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html*

[4] An electronic version of part of this glossary can be found in *http://turing.iimas.unam.mx/c̃rodriguezp/gloss/index.htm*

The evaluation protocol was inspired by [6]. Judges scored between 1 and 5 each test classification. Score 1 means that the cluster is non-sensical (and so term classification). Score 2 means that the term was oddly classified in a correct cluster. 3 means the cluster is correct but the evaluator is undecided about the correctness of classification. 4 means the term fits with the general sense of the cluster (which is correct). Finally, 5 means the term fits well with the cluster (which must be correct).

Table 4 illustrates the results of evaluation. Note that most clusters are meaningful since only about 5% of classifications are non-sensical (Classif. 1).

Table 4. Evaluation of Word Classification

	Judge_1	**Judge_2**	**Judge_3**
Classif. 1	3.94%	3.82%	8.12%
Classif. 2	7.89%	5.73%	11.25%
Classif. 3	12.50%	12.10%	17.50%
Classif. 4	21.05%	26.11%	10.62%
Classif. 5	54.60%	52.22%	52.50%
Number of Tests	160		
Average Score	4.10		
Average Difference	0.15		

In further research, we intend to develop a process of context/property classification using the classes learnt by means of the specification operation. In this process, each specific class will be assigned unknown lexico-syntactic contexts that were not involved in the previous clustering steps.

References

1. P. Allegrini, S. Montemagni, and V. Pirrelli. Example-based automatic induction of semantic classes through entropic scores. *Linguistica Computazionale*, pages 1–45, 2003.
2. Paolo Allegrini, Simonetta Montemagni, and Vito Pirrelli. Learning word clusters from data types. In *Coling-2000*, pages 8–14, 2000.
3. David Faure and Claire Nédellec. Asium: Learning subcategorization frames and restrictions of selection. In *ECML98, Workshop on Text Mining*, 1998.
4. Pablo Gamallo, Alexandre Agustini, and Gabriel Lopes. Clustering syntactic positions with similar semantic requirements. *Computational Linguistics*, 31(1):107–146, 2005.
5. J. Hereth, G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery - a human-centered approach. *Journal of Applied Artificial Intelligence*, 17(3):288–301, 2003.
6. Dekang Lin and Patrick Pantel. Induction of semantic classes from natural language text. In *SIGKDD-01*, San Francisco, 2001.
7. Patrick Pantel and Dekan Lin. Discovering word senses from text. In *ACM SIGKDD*, pages 613–619, Edmonton, Canada, 2002.
8. Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *ACL'93*, pages 183–190, Columbos, Ohio, 1993.
9. Uta Priss. Formal concept analysis in information science. *Information Science and Technology*, 40:521–543, 2006.

10. M-L. Reinberger and W. Daelemans. Is shallow parsing useful for unsupervised learning of semantic clusters? In *CICLing'03*, pages 304–312, Mexico City, 2003.
11. Mats Roth. Two-dimensional clusters in grammatical relations. In *AAAI-95*, 1995.