Towards the derivation of verbal content relations from patent claims using deep syntactic structures

Gabriela Ferraro a,*, Leo Wanner b

aDept. of Communication and Information Technologies, Pompeu Fabra University, Spain
bCatalan Institution for Research and Advanced Studies, Pompeu Fabra University, Spain

ABSTRACT

Research on the extraction of content relations from text corpora is a high-priority topic in natural language processing. This is not surprising since content relations form the backbone of any ontology, and ontologies are increasingly made use of in knowledge-based applications. However, so far most of the works focus on the detection of a restricted number of prominent verbal relations, including in particular IS-A, HAS-PART and CAUSE. Our application, which aims to provide comprehensive, easy-to-understand content representations of complex functional objects described in patent claims, faces the need to derive a large number of content relations that cannot be limited a priori. To cope with this problem, we take advantage of the fact that deep syntactic dependency structures of sentences capture all relevant content relations—although without any abstraction. We implement thus a three-step strategy. First, we parse the claims to retrieve the deep syntactic dependency structures from which we then derive the content relations. Second, we generalize the obtained relations by clustering them according to semantic criteria, with the goal to unite all sufficiently similar relations. Finally, we identify a suitable name for each generalized relation. To keep the scope of the article within reasonable limits and to allow for a comparison with state-of-the-art techniques, we focus on verbal relations.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

The last years saw a significant increase in research on content mining in text corpora. One aspect of this research targets the extraction of content relations, mostly for the construction or extension of ontologies [48,11,12]. We, on our part, are interested in the extraction of content relations encountered in the descriptions of complex functional objects, i.e., devices. The corpus on which we focus is a corpus of patent claims; the devices are thus illustrations. However, the extraction of relations between the individual components of an object visualized in block diagrams or concept maps implies a number of challenges not faced in the construction of ontologies. As illustrated by (1), in patent claims content relations between components of an object are expressed by verbs, predicative nouns and adjectives. Consider, for example,
device COMPRISING a lens, PLURALITY of light receiving sections, CENTRAL portion of the light receiving surface, etc. A straightforward use of these terms as labels of content relations in a block diagram (as done in [9]) or in a concept map drawing is not appropriate: block diagrams and concept maps are conceptual representations. They must achieve abstraction over concrete terms. For instance, the terms COMPRISE, HAVE, CONTAIN, CONSIST OF and the like should be all subsumed by the concept ‘part of’; the terms CAUSE, LEAD TO, RESULT IN, etc. should be captured by the concept ‘cause’; the terms MODIFY, CHANGE, ALTER, etc. may be well represented by the concept ‘modify’; and so on. For the sake of more uniform and simple knowledge representations, the relation abstraction should go even farther. Thus, we should also aim at finding a common conceptual equivalent for more heterogeneous (but still sufficiently semantically similar) terms such as ADJUST, ARRANGE, PLACE, POSITION, and SET, OR ENCODE, ENCRYPT, INSCRIBE, etc.—just to give a few examples. However, we cannot deduce from this need for abstraction that the number of relations can be limited to a predefined set: Although knowledge representations are better understood if they operate with a small number of relations, the number and type of these relations cannot be defined a priori without drastically limiting the application domain. This makes our work different from the known state-of-the-art relation extraction techniques, which focus on a restricted number of relations between nominals, as [2,40,23], or which do not abstract from specific relations at all, as [13,20].

In order to provide adequate content relations as input to block diagrams or concept map drawing programs, the goals of our work are threefold:

1. Extract content relations between components of a functional object described in a patent claim.
2. Generalize over similar content relations by clustering them according to semantic criteria.
3. Find an appropriate (concept) label for each cluster, i.e., generalized relation.

For the generalization task, we use an external semantic resource, the lexical database EuroWordNet [58], which has already been used frequently in text mining.

To keep the scope of the article within reasonable limits, we shall restrict our discussion to the description of the extraction of verbal content relations and to their subsequent clustering and labeling. The remainder of the article is structured as follows: Section 2 reviews the existing works in the relevant research areas. In Section 3, we analyze the problem of content relation extraction and clustering from the viewpoint of linguistic codification. The results of this analysis help us formulate our proposal for how the problem can be approached from the computational side. Section 4 describes our approach to automatic content relation extraction, clustering (i.e., grouping) and labeling. Section 5 contains an evaluation of the implementation of the individual stages of our approach. Section 6 discusses the results obtained during the evaluation. Finally, Section 7 contains a short conclusion drawn from our experience and outlines the direction of future work.

2. Related work

Let us briefly review the related work in the three areas with which we are concerned: content relation extraction, content relation clustering, and cluster labeling.

2.1. Related work on content relation extraction

In the past, relation extraction has often been addressed by drawing upon one of the following three sources of linguistic information: 1. lexico-syntactic patterns, 2. purely syntactic patterns, and 3. co-occurrences of words by which the relations in question are expressed.

Lexico-syntactic patterns of the kind (X) such as (Y), by such (X as Y), (X including Y), etc. are certainly the most common source, especially when the set of relations is restricted and identified beforehand. The patterns are determined either manually, as in [2,6,40], or derived from a corpus using machine learning techniques, as in [47,1,48,44,23,7]. In some works, the lexico-syntactic patterns are enriched by name entity tags or WordNet features [21] to extract such relations as LOCATION-ORGANIZATION, NEAR, etc.; see e.g., [1,25]. Girju et al. [23] enrich relation patterns using hypernymy features from WordNet to distinguish between correct and incorrect PART-OF relations. Other extensions of the pattern-based relation extraction include filters for selecting only the most prominent patterns; see, among others, [3,20].

Purely syntactic patterns are more flexible than lexico-syntactic patterns in that they facilitate the extraction of any semantic type of relation. Both dependency patterns and constituency patterns have been used. Ciaramita et al. [11] extract (subject, verb, object) and (subject, verb, indirect object) pattern instances, which are then filtered for significance using the χ² test. Cimiano et al. [13] extract (NP₁ – V – NP₂) and (NP₁ – V – PP – NP₂) pattern instances and from these derive relations with the Vs as their labels and the heads of NP₁ and NP₂ as their arguments.

An important amount of work on relation extraction is based on word co-occurrences in a restricted context. In general, co-occurrence identification techniques are based on n-gram frequency and statistical association measures. For instance, Yamaguchi [60] learns non-taxonomic relations using high frequency 4-gram co-occurrences. Co-occurrence-oriented relation extraction via supervised machine learning techniques has been recently also explored at length in such competitions as ACE [18] and SemEval [24]. Both offer a corpus tagged with a predefined set of relations for training. In ACE, the following five relations are addressed, some of which are further subclassified so that a total of 24 relations are tagged: 1. ROLE (the role a person plays in an organization), 2. PART (the part of a whole), 3. AT (location), 4. NEAR (relative location), and 5. SOCIAL (cognition relations). Consult, for instance, [15,43,51] for different proposals in the ACE-competition.


2.2. Related work on relation clustering

Most of the recent works on the semantic classification of verbs draw upon the verb taxonomy of Levin [33] using supervised and unsupervised; see [6,59,31,50,49,57]. Somewhat simplified, this taxonomy captures the projection of semantic valency structures of verbs onto their syntactic valency structures. Since our goal is a grouping (or clustering) based on purely semantic criteria, we make no use of Levin’s taxonomy.

A number of works focus on the clustering of (already extracted) relation instances—for instance, using a semantic distance measure [10,63]. Many proposals have been made as to how the similarity distance between lexical units (LUs) may be calculated. One way to know how similar or different an LU is with respect to another LU is to measure the similarity of their context vectors in a Vector Space Model (VSM) [45].

Language
way is to use Wordnet’s hierarchical structure by comparing the hyperonymy chains of the LUs in question—as suggested, e.g., by Yang and Powers; see [61] for their proposal for measuring semantic similarity between nouns and [62] for measuring semantic similarity between verbs. The latter adapts the first method, although with a limited success.

2.3. Related work on cluster labeling

Cluster labeling deals with finding an appropriate name for a given group (= cluster) of similar elements. Two main strategies can be distinguished in the literature: (i) internal cluster labeling and (ii) differential cluster labeling. In the first, the label for a given cluster is chosen drawing solely on the cluster itself. In the second, the label for a cluster is chosen by contrasting this cluster with the other available clusters.

The proposal by Pantel and Ravichandran [41], which addresses the problem of labeling clusters of semantically similar nouns, is an example for internal cluster labeling. As already some works on relation extraction, it draws upon syntactic patterns: First, for each element of a cluster “grammatical signatures” that capture its prototypical syntactic context in different occurrences are computed. Then, among these signatures, simple hypernym-patterns, such as (noun–apposition–noun) (for instance, H1N1, the disease, which . . .) and (noun such as noun) (for instance, diseases such as H1N1) are searched. At last, the mutual information scores for each hypernym candidate are calculated and the highest scoring hypernym is chosen as the name of the cluster. Further similar proposals include [8,35].

The proposal by Dias et al. [16], which addresses the problem of clustering of webpage results and the subsequent labeling of the obtained clusters, is an example for differential cluster labeling. It chooses as label of a cluster a noun or a noun compound that (i) occurs in most of the URLs of the cluster in question, (ii) discriminates the cluster sufficiently well from the other clusters. Other proposal for differential cluster labeling is, for instance, [55,19].

3. Codification of content relations in language

As already pointed out, the goal to capture all content relations that contribute to the description of a complex functional object implies that our task cannot be restricted to a limited number of relations, and, thus, to the use of lexico-syntactic patterns for relation detection. Nor can we rely upon the co-occurrence of words that engage in a relation since the idiosyncratic style of patent material leads to the distribution of these words over a long distance (consider claim (1) above, where the arguments of comprising (in An automatic focusing device comprising: . . .) appear at a distance of 22, 56, 76, 167, and 203 tokens with respect to comprising itself). But the use of syntactic constructions seems to be an option. For greater clarity, however, we need to know how content relations in patent claims are encoded in syntactic terms and how the semantic similarity of labels that express content relations can be assessed.

3.1. Codification of content relations

As already mentioned in Section 1, a content relation can be rendered in a text by a verb, predicative noun or an adjective. More precisely, a content relation can be considered a relation of the type ⟨‘label’ Ai, A2, A3, . . .⟩, where ‘label’ is the relation’s name with the morpho-syntactic category ‘verb’, ‘noun’ or ‘adjective’ and Ai (i = 1,2, . . .) are the arguments of the relation. In linguistic terminology, Ai fills a valency slot of the label.

Each predicative lexeme possesses a valency structure. The valency of a lexeme L specifies how many arguments (or actants) L possesses. For instance, STOP1 has one argument: ‘X stops’ (as in The recording device stops when it reaches the end of the recording medium); COMPRISE1 has two arguments: ‘X comprises Y’ (as in The device comprises a lens); and SEPARATE1 has three arguments: ‘X separates Y from Z’ (as in The device separates the husk from the corn). To extract a content relation instance labeled by L from a corpus, we need thus to identify the argument instances of L and to associate them with the specific slots in the valency structure of L. Only then will we be able to obtain the basis for a conceptual representation (thus, the surface reflects the light is not the same as the light reflects on the surface). For this purpose, the use of syntactic dependency structures of Ls seems promising since dependency structures represent, in fact, a projection of (semantic) predicate-argument structures and thus automatically capture the arguments of Ls. The condition is that we do not use use surface-oriented dependency structures, which capture grammatical functions of the kind as the left structure in Fig. 1 (although most dependency parsers deliver them), but abstract syntactic structures of the type of deep-syntactic structures (DSyntSs) in Meaning-Text Theory [36], such as the right-side structure in Fig. 1.

A DSyntS is a dependency tree with nodes labeled by “deep” lexical units and arcs labeled by universal syntactic relations: actants or arguments (I,II,III, . . .), attribute (ATTR), apposition (APOS), and coordination (COORD). Each of the actantial relations I, II, III, etc. relates the predicative lexeme L with the respective argument instance, while L functions as a label that relates the arguments among each other. The attribute relation ATTR relates a lexeme to a modifier, which can be an adjectival phrase, an attributively used verbal phrase, or a prepositional phrase that acts as modifier of space, time, etc. In contrast to the case of argument relations, no modifier label can be searched for in the text. This label must be deduced via an interpretation of the function or of the semantics of the modifier in question: member ✏ kind → chunking, part ✏ position → central, surface ✏ color → black, etc.

3.2. Assessing the similarity between content relations

While abstract syntactic structures are helpful for the detection of the instances of content relations, they still do not suffice. As argued in Section 1, it is not appropriate to distinguish between such concrete verbal or nominal relation labels as construct and create, emit and radiate, expose and show, etc. for use in conceptual repre-
sentations. Rather, a semantic abstraction should be sought that helps reduce the number of labels, selecting labels that each subsume a number of sufficiently similar concrete labels.

The first step towards the abstraction of labels is to determine the similarity between given labels. One way to do this is to use a lexical database such as WordNet [21] or EuroWordNet [58] as an external source. Both have been repeatedly used in previous works as a source of fine-grained semantic information—also in the context of semantic lexeme clustering (Section 2). Since the differences between both are not of relevance here, let us refer henceforth to both as ‘WN’.

WN follows the relational paradigm of the lexicon, which means that the meaning of a given lexeme (which is a word in one of its meanings) is specified in terms of its relations to other lexemes. The most important relations are hyperonymy, hyponymy, synonymy, antonymy, holonymy and meronymy. For instance, in WN the lexemes \texttt{DISPLAY1} and \texttt{DISPLAY2} obtain the description of hyperonym and synonym relations as presented in Table 1, and \texttt{SHOW1} and \texttt{SHOW2}, the description shown in Table 2.

Comparing the descriptions of both \texttt{DISPLAY1}, \texttt{DISPLAY2}, \texttt{SHOW1}, and \texttt{SHOW2} we can thus deduce with a considerable accuracy how similar or different the meanings of these two lexemes are. As mentioned in Section 2, hyperonym chains from WN have been used to determine the similarity between nouns. Verbal hyperonym chains are much less extended in WN than nominal hyperonym chains. This makes their use as semantic descriptors much less reliable; Yang and Powers [62], already mentioned in Section 2, had a similar experience. Therefore, we use synonymy instead of hyperonymy, representing each verbal relation label as a vector of its WN-synonyms.\footnote{\textit{In WN, the synonyms of a lexeme are summarized in a synset.}}

Consider, for illustration, the vectors of \texttt{CONSTRUCT1}, \texttt{CREATE6}, \texttt{MAKE24}, and \texttt{PRODUCE2}, which are intuitively semantically rather close:

\begin{itemize}
\item \texttt{CONSTRUCT1}: \{construct1, build1, make17\}
\item \texttt{CREATE6}: \{create6, produce2, make6\}
\item \texttt{MAKE24}: \{make24, build1, construct1\}
\item \texttt{PRODUCE2}: \{produce2, make6, create6\}
\end{itemize}

\begin{table}[ht]
\centering
\caption{WN description for \texttt{DISPLAY1} and \texttt{DISPLAY2}.}
\begin{tabular}{ll}
\hline
\texttt{DISPLAY1} & \texttt{DISPLAY2} \\

\hline
\textbf{Definition} & To show, make visible or apparent \\
\textbf{Synonyms} & expose3, exhibit2 \\
\textbf{Hyperonyms definition} & Make visible or noticeable \\
\textbf{Hyperonyms} & show4 \\
\hline
\end{tabular}
\end{table}

\begin{table}[ht]
\centering
\caption{WN description for \texttt{SHOW1} and \texttt{SHOW2}.}
\begin{tabular}{ll}
\hline
\texttt{SHOW1} & \texttt{SHOW2} \\

\hline
\textbf{Definition} & Show or demonstrate something to an interested audience \\
\textbf{Synonyms} & demo1, exhibit2, present1, demonstrate1 \\
\textbf{Hyperonyms definition} & Make visible or noticeable \\
\textbf{Hyperonyms} & show4 \\
\hline
\end{tabular}
\end{table}

Vectors of semantically close lexemes can thus be assumed to overlap with a certain probability—although not always. Observe, for instance, \texttt{CONSTRUCT1}: \{construct1, build1, make17\} and \texttt{CREATE1}: \{create1, produce2, make6\}, and for \texttt{DIFFERENTIATE1} \{differentiate1, distinguish1, seern1, seernate1\} and \texttt{DISTINGUISH1} \{distinguish1, seern1, seernate1\}, where this is not the case. But the transitivity of semantic similarity will still often allow us to establish a semantic link between them. This is not to say that we will always succeed in grouping semantically similar relation labels by calculating the similarity of their synset-vectors. For this, the word sense distinction in WN is too fine-grained and sometimes too \textit{ad hoc}.

4. Model for retrieving content relations

Taking the findings of the previous section into account, we adopt the following generic three-stage procedure to retrieve content relations from patent claims:

1. distill the relations via deep dependency parsing.
2. cluster the relation tuples obtained in the previous stage according to WN’s synonymy criteria.
3. assign to each cluster an appropriate label.

4.1. Distilling relations via deep dependency parsing

In accordance with the discussion in Section 3, we divide the first stage into three sub-stages:
(a) obtain the deep-syntactic structure $D_S$ of each sentence $i (i = 1, \ldots, n)$ from which the content relations are to be distilled,
(b) extract the dependency relations of each lexeme from each $D_S$,
(c) convert the dependency relations into relation tuples.

4.1.1. Obtaining deep-syntactic structures

Off-the-shelf dependency parser technologies cannot be used as such to perform deep parsing of patent claim material in order to obtain deep-syntactic structures ($DSyntS$s). This is because: (i) they tend to provide surface syntax-like structures rather than $DSyntS$s as their output; (ii) they perform poorly even for surface syntax structures due to the length and complexity of the patent claim sentences. Therefore, in order to obtain the $DSyntS$s from which we can distill argument structures that can then be converted into content relations, we apply a three step procedure: (a) simplify the original sentences; (b) parse the simplified sentences; (c) map the parser output onto $DSyntS$s.

Preprocessing: simplification of patent claims. The objective of the simplification, which is described in detail in [5], is to improve parsing without any loss of relevant linguistic information. As the illustration in Fig. 2 shows, the simplification involves the following steps:

1. part-of-speech (POS) tagging and chunking using the TreeTagger [46] with its off-the-shelf English parameter set;
2. segmentation of the claim sentence into clausal discourse units using a machine learning-based segmentation algorithm that uses lexical, punctuation, POS and chunking information;
3. establishment of the coreference links between NPs that denote the same object;
4. building a clause-discourse tree by drawing upon coreference links and other information such as tree configuration criteria and discourse markers;
5. reconstruction of the individual clauses in order to obtain a grammatically correct independent sentence out of each of them.

When applied to the original claim in (1), the simplification provides the result in (2):

(2) An automatic focusing device comprising: an objective lens; a beam splitter; an astigmatic optical system; a light detector; a focal position driving circuit. The objective lens focuses light beam. The light source emits a light beam on a track of an information recording medium. The beam splitter separates the reflected light beam. The information recording medium reflects the reflected light beam at a focal spot thereon and through the objective lens from the light beam. The light source emits the light beam. The astigmatic optical system includes an optical element. The optical element is capable of causing the astigmatic aberration of the separated reflected light beam. The light detector has a light receiving surface. The light receiving surface is divided, except the central portion thereof, into a plurality of light receiving sections. The light receiving sections are arranged symmetrically with respect to a first axis and to a second axis. The first axis extends in parallel to the axial direction of the optical element. The second axis extends perpendicularly to the first axis. The second axis is adapted to receive the reflected beam. The reflected beam is transmitted through the optical element. The second axis is adapted to give a light reception output signal corresponding to the shape of the spot of the reflected light beam. The reflected light beam is formed on the light receiving surface. The focal position detecting circuit capable of giving an output signal corresponding to the displacement of the objective lens from the focused position, on the basis of the output signal given by the light detector. The lens driving circuit drives the objective lens along the optical axis on the basis of the output signal given by the focal position detecting circuit.

The most obvious differences between (1) and (2) are that in (2) nearly each clause forms a sentence, and the arguments of the predicative lexemes are reordered to be at the minimal distance from the head.

Parsing simplified sentences The simplified sentences are parsed using the off-the-shelf dependency parser MiniPar [34]. MiniPar has been chosen because it produces syntactic structures that are in their nature similar to surface-syntactic structures ($SSyntS$s) in MTT (both capture grammatical functions of the lexemes in a sentence). Furthermore, it is robust and performative enough for our task. Consider a sample structure obtained as output from MiniPar in Fig. 3.

On the basis of the information provided by MiniPar, we thus obtain $DSyntS$s in two steps [37]: First, MiniPar structures are mapped onto MTT’s $SSyntS$s, and then $SSyntS$s are mapped onto $DSyntS$s (consider Fig. 3 for the $DSyntS$ that corresponds to the MiniPar structure presented to the left).

Despite their similarity, MiniPar output structures and $SSyntS$s show some crucial differences, which are due to the theoretical divergences of MTT and the linguistic model underlying MiniPar. This makes the MiniPar-to-$SSyntS$ projection less than straightforward. The current version of the MiniPar-$SSyntS$ mapping grammar contains 137 rules [38]. Its evaluation on 1324 sentences...
has shown that 99% of well-formed MiniPar-structures are correctly mapped onto SSyntSs.

4.1.2. Obtaining relation tuples

Once the DSyntS are obtained, we first extract from them the dependency relations of the individual lexemes. For instance, for the DSyntS in Fig. 3 we get: extend-I → axis, extend-II → parallel, parallel-I → axis, parallel-II → direction, axis-ATTR → first, direction-ATTR → axial, direction-I → element, el-ement-ATTR → optical. These relations are then converted into content relation tuples. During the conversion, the following actions are performed:

(i) attribute relation markers between object denoting nouns (such as lens and component) and their modifiers are eliminated and the heads and their modifiers are concatenated to form multiform terms; for instance, direction-ATTR → axial becomes direction and element-ATTR → optical becomes optical element;

(ii) the argument relation markers between heads and their (concatenated) arguments are eliminated and the arguments are ordered relatively to their head in the prefix notation; thus, extend-I → first axis becomes (extend)(first axis), extend-I → parallel becomes (extend)(-,parallel), and parallel-II → axial direction becomes (parallel)(-,axial direction);

(iii) overlapping pairs of relations are aggregated into one relation: (extend)(first axis), (extend)(-,parallel), and (parallel)(-,axial direction) become (extend)(first axis),(parallel)(-,axial direction)).

Consider the verbal relations extracted from the simplified claim (2); for later reference, they are indexed.

(3) comprise$_1$(automatic focusing device, objective lens), comprise$_2$(automatic focusing device, beam splitter), comprise$_3$(automatic focusing device, astigmatic optical system), comprise$_4$(automatic focusing device, focal position detecting circuit), comprises$_5$(automatic focusing device, lens driving circuit), focus$_6$(objective lens, light beam), emit$_7$(light source, light beam), track of an information recording medium), separate$_8$(beam splitter, reflected light beam), reflect$_9$(information recording medium, light beam, focal spot, object lens, include$_{10}$(slf astigmatic optical system, optical element), be-capable$_{11}$(optical element, cause$_{12}$(optical element, astigmatic aberration)), have$_{13}$(light detector, light receiving surface), divide$_{14}$(light receiving surface, plurality of light receiving sections), arrange$_{15}$(light receiving sections, symmetrically, first axis and second axis), extend$_{16}$(first axis, parallel, g(,-axial direction)), extend$_{17}$(second axis, perpendicularly,-g(,-first axis)), be-adapted$_{18}$(second axis, receive$_{19}$(second axis, transmit$_{20}$(optical element, reflected beam)), be-adapted$_{21}$(second axis, give$_{22}$(second axis, light reception output signal)), correspond$_{23}$(light reception output signal, spot shape), form$_{24}$(second axis, light receiving surface), capable$_{25}$(focal position detecting circuit, give$_{26}$(focal position detecting circuit, output signal)), drive$_{27}$(lens driving circuit, objective lens).

4.2. Relation clustering

Once the relation tuples have been distilled, their names are clustered in accordance with their semantic similarity. This is done in order to generalize the relations as much as possible (and to find the most adequate labels in the next stage). The clustering procedure consists of two steps. In the first step, the similarity between the collected relation names is calculated. In the second step, the names are clustered according to this similarity.

In a generic setting, we must know in which of its senses a word is used in order to be able to assess its similarity to another word. Thus, to compare [to] separate with [to] divide as they appear in patent claims, we must know that it is the separate in the sense of ‘to bring physically apart’ rather than in the sense of ‘to part company’, which is of relevance. In other words, we would need to apply word sense disambiguation, which is a rather complex task.

However, extending the one-sense-per-discourse assumption [22] to the one-sense-per-patent-domain assumption we may hypothesize that we can largely dispense with word sense disambiguation. That is, we can assume that in a specific patent domain, a given word is mainly used in the same sense. For instance, [to] separate in patents of the domain of Machine Tools will in the majority of the cases mean ‘action of bringing parts physically apart’, and [to] record in Optical Recording Devices patents will mean ‘action of registering information on a digital medium’. Therefore, if there is a sense of the verb V that is similar to a sense of the verb W, we will detect this similarity by measuring the similarity of all senses of V with all senses of W (see Section 5 for a verification of this assumption). For this purpose, we construct a synonomy vector for each sense $v_i$ of V and for each sense $w_j$ of W captured in WN: $V_i=(v_{i1}, v_{i2}, \ldots, v_{in})$, with n as the number of synonyms for $v_i$ and $w_j=(w_{j1}, w_{j2}, \ldots, w_{jm})$ with m as the number of synonyms for $w_j$. The similarity between V and W is then assumed to be the maximum similarity between any sense $v_i$ of V and any sense $w_j$ of W, calculated as the maximum vector cosine:

\[
\text{Sim}(V, W) = \max \text{cos}(\vec{v}_i, \vec{w}_j), \quad i=1, \ldots, k \quad (k \text{ as the number of senses of } V) \quad j=1, \ldots, p \quad (p \text{ as the number of senses of } W). \tag{1}
\]

\[
\text{cos}(\vec{v}_i, \vec{w}_j) = \frac{\vec{v}_i \cdot \vec{w}_j}{|\vec{v}_i| \cdot |\vec{w}_j|} = \frac{\sum_{s=1}^{N} v_{is} w_{js}}{\sqrt{\sum_{s=1}^{N} v_{is}^2} \sqrt{\sum_{s=1}^{N} w_{js}^2}}
\]

(With $N$ as the normalized length of the synonymy vectors).

For instance, according to WN, the verb DISPLACE has six senses and the verb MOVE 14 senses; Fig. 4, which displays the senses for DISPLACE and MOVE with their corresponding sets of WN synonyms.

The highest cosine value (and thus the greatest similarity) turns out to be between the 4th sense of DISPLACE and the 2nd sense of MOVE.

The similarities obtained between the relation names are most conveniently represented in terms of a similarity matrix, as shown in Table 3. Each row $i$ / column $j$ of this matrix represents a relation label, and the value in the cell $(i,j)$ is the similarity between label $i$ and label $j$.

With the similarity matrix at hand, we can cluster the relations in accordance with their similarity. For this purpose, we apply Cluto to clustering algorithms [28].

So far, we have experimented with three different algorithms which appeared most promising: (i) partitional clustering, (ii) agglomerative clustering, and (iii) graph-partitioning-based clustering. Each of these implements a different clustering paradigm. A variant of graph-partitioning-based clustering performed best: the optimized repeated bisections (ORB) algorithm [64]. The ORB algo-

---

8 Cosine has been chosen because it is the most common measure for calculating similarity of decomposed semantic descriptions of lexical units: it is rather simple and reliable.

9 Cluto offers a standard implementation of the major clustering algorithms, covering both the hierarchical and the partitional paradigms of clustering. A further advantage of Cluto is its efficiency: as most high-dimensional datasets, the resulting similarity matrix is sparsely populated. Cluto takes into account this sparsity and requires memory that is roughly linear in relation to the input size.
A clustering algorithm divides a given collection of entities into \( k \) clusters by performing a sequence of \( k/C_0 \) bisections. First, the whole collection is divided into two clusters. Then, one of these clusters is selected and bisected. And so on, until the desired number of clusters (\( k \)) is reached. Each bisection and the overall solution are optimized using the \( H_1 \) criterion function:

\[
H_1 = \text{maximize } \frac{\sum_{r=1}^{k} ||D_r||^2/n_r}{\sum_{r=1}^{k} n_r D_r ||D||},
\]

(2)

where \( D_i \) is the composite vector of the cluster number \( r \), \( n_r \) is the size of \( r \), \( k \) is the number of clusters, \( D_r^t \) is the composition of the cluster \( r \), and \( D \) is the composite vector of the entire relation label collection.

As shown in (2), \( H_1 \) is a hybrid criterion function, which combines the internal criterion function \( I_1 \) and the external criterion function \( E_1 \); while \( I_1 \) produces a clustering solution that optimizes a function defined over the elements that are part of a cluster (not taking into account the elements assigned to other clusters), \( E_1 \) takes into account the difference between the different clusters, trying to separate the elements of each cluster from the entire collection.

We experimented with different values for \( k \), with the goal to find a balance between cluster purity and the number of singleton clusters. Purity is reciprocally proportional to the number of singleton clusters. In particular, purity is \( 1 \) if each element is assigned its own cluster—which is, obviously, not our goal. Fig. 5 shows the results of the trials with \( k \) from 35 to 80. According to these trials, the best \( k \) is around 60. This means that given 321 verbal relation labels in our experimental collection, each cluster will ideally contain about five labels. We think that this reflects the structure of the data in a conservative approach to our task since in order to keep the expressiveness of the relation labels, we do not want them to be too abstract (which would be the consequence of large clusters).

Some sample clusters are given in Table 4.

The most homogeneous among the obtained clusters is \{displace, impress\}. Only one member of this cluster is not well grouped: impress. The reason is that in WN one of the senses of move stands for ‘emotional or cognitive impact’.

### 4.3. Cluster labeling

We explored three different strategies for cluster labeling, i.e., for finding a common abstract label for a set of semantically similar concrete (verbal) relation labels. In all three of them we computed a label that depends only on the cluster itself and not on other clusters. This means that all three strategies are cluster-internal.

**Frequency-oriented labeling** (Freq). This strategy chooses as the cluster label the member of the cluster with the highest frequency in the corpus. Thus, for the cluster \{displace, impress, move\}, this strategy would suggest move as cluster label.

**Similarity-oriented labeling** (Sim). This strategy chooses as the cluster label the member of the cluster with the highest similarity to the other members of the cluster. Thus, for the cluster \{displace, impress, move\}, this strategy would suggest impress as cluster label.

**Distance-oriented labeling** (Dist). This strategy chooses as the cluster label the member of the cluster with the lowest distance to the other members of the cluster. Thus, for the cluster \{displace, impress, move\}, this strategy would suggest move as cluster label.
context of finding a common label for semantically similar yet still different lexemes, it seems more intuitive to select an abstract label that represents all members of the cluster.

Verb hyperonym-oriented labeling (VHyper). This strategy chooses as the cluster label the most frequent hyperonym of the cluster as it appears in the WN verb hierarchy. Only hyperonyms of verb senses with non-zero similarity (see Section 4.2) are taken into account. First, for each member of the cluster, all its WN hyperonyms are retrieved and the most frequent hyperonym set is selected. From this set, the most frequent lexeme is chosen as the cluster label. Consider Table 5 for illustration where the first column shows the hyperonym set in this case is:

bound3, check4, confine1, limit1, restrain2, restrict3, throttle1, trammel2, decide1, decide upon1, determine4, make-up one’s-mind

The most frequent lexeme in our corpus from this hyperonym set is LIMIT1. LIMIT is therefore chosen as cluster label.

This strategy is motivated by the fact that the cluster label should be more abstract in order to ensure that all members of the cluster are reflected.

Noun hyperonym-oriented labeling (NHyper). This strategy chooses the most frequent hyperonym among the nominalized members of a cluster as the cluster label by drawing upon the WN noun hierarchy. This is possible because some verbs are derived from nouns, or vice versa, such that the WN noun hierarchy can be used to explore the relations between verbs [62].

The motivation behind this strategy is that (as already mentioned before) while for verbs in WN only relatively flat hyperonym hierarchies are available, nominal hyperonym hierarchies tend to be rich and deep. Again, only hyperonyms of the verb senses with a non-zero similarity are taken into account.

However, after experimenting with this strategy, we decided to discard it because it requires substantial morphological preprocessing (at least stemming and morphological derivation analysis) to achieve good quality.

Table 6 displays the results of the application of the two first labeling strategies to a number of manually compose clusters (so-called gold standard clusters). ‘GS’ stands for “gold standard labels”, i.e., labels manually assigned to the corresponding clusters.

![Fig. 5. Clustering experiments with different kS, purity and number of singleton clusters.](image-url)
Examples of the performance of the Freq and VHyper labeling strategies with manually compiled clusters.

<table>
<thead>
<tr>
<th>Gold Standard Clusters</th>
<th>GS</th>
<th>Freq</th>
<th>VHyper</th>
</tr>
</thead>
<tbody>
<tr>
<td>(comprise, contain, have, include)</td>
<td>contain</td>
<td>comprise</td>
<td>comprise</td>
</tr>
<tr>
<td>(bound, limit, restrain, inhibit)</td>
<td>limit</td>
<td>inhibit</td>
<td>determine</td>
</tr>
<tr>
<td>(tighten, fasten, fix, secure, deposit)</td>
<td>fix</td>
<td>fix</td>
<td>fix</td>
</tr>
<tr>
<td>(compress, trim, reduce, minimize)</td>
<td>reduce</td>
<td>reduce</td>
<td>cut</td>
</tr>
<tr>
<td>(extract, pull-out)</td>
<td>extract</td>
<td>extract</td>
<td>remove</td>
</tr>
<tr>
<td>(remove, cut, delete, erase, exclude)</td>
<td>remove</td>
<td>remove</td>
<td>remove</td>
</tr>
<tr>
<td>(enter, insert, interpose, introduce, enclose)</td>
<td>insert</td>
<td>insert</td>
<td>connect</td>
</tr>
<tr>
<td>(apply, feed, provide, give, use, supply, render)</td>
<td>produce</td>
<td>provide</td>
<td>provide</td>
</tr>
<tr>
<td>(hold, maintain, retain, support, prevent)</td>
<td>keep</td>
<td>support</td>
<td>maintain</td>
</tr>
<tr>
<td>(accord, allow, let, permit)</td>
<td>let</td>
<td>accord</td>
<td>have</td>
</tr>
</tbody>
</table>

Table 6 presents some examples of the performance of Freq and VHyper for clusters composed automatically.

It is interesting to observe that even when the clusters contain outliers, such as (tighten, fasten, fix, secure, deposit), where the inclusion of deposit is clearly a clustering error, the labeling strategies were not seriously affected when the number of errors per cluster was low.

For our sample patent claim in (1) and (2), the clustering and the subsequent cluster labeling leads to a generalization over the following relations listed in (3):

- focusc → point: point(objective lens, light beam)
- reflectc → emit: emit(information recording medium, light beam, focal spot, objective lens)
- includex1 → comprise: comprise(astoigmatric optical system, optical element)
- havex1 → comprise: comprise(light detector, light receiving surface)
- arrangeix1 → set: set(=, light receiving sections, symmetrically, first axis and second axis)
- formy27 → perform: perform(=, reflected light beam, light receiving surface)

Two of these generalizations (namely reflectc → emit and formy27 → perform) appear questionable as they considerably change the semantics of the original relation.

### 5. Evaluation

In what follows, we evaluate the three central stages of our content relation extraction and clustering procedure: verbal extraction, clustering, and cluster labeling. We refrain from the evaluation of claim simplification, which constitutes our pre-processing stage, since it has already been evaluated in detail in [5]. An evaluation of MiniPar on the SUSANNE corpus can be found in [34].

#### 5.1. Evaluation of the verbal relation extraction stage

The goal of the evaluation of this stage is to assess how well our model is able to distill relations of the type (arg1) VERB (arg2) from patent claims. For the purpose of a comparative evaluation, we applied our verbal relation extraction procedure and then the relation extraction procedure proposed by Cimiano et al. [13] to the claims of four European patents (EP0007902A1, EP0548937A1, EP0067095A2, and EP0080884A2) first. We have chosen Cimiano et al.’s reference because it aims to extract all the verbal relations available in a text, as we do. From the existing 94 verbal relations annotated as gold standard, our procedure correctly extracted 67 (which amounts to 71%), and Cimiano et al.’s procedure 51 (54%).

#### 5.2. Evaluation of clustering

For the evaluation of clustering and cluster labeling, we used a corpus of 2076 English patent claims (the reference numbers of the corresponding patents are listed in an Appendix). From this corpus, a list of the 321 most frequent verbs was compiled (verbs with less than three occurrences were dropped).

The verbs have been first clustered manually to obtain the gold standard (reference) clusters (54 in total) by two annotators not involved in the work otherwise. The clustering procedure consisted of two rounds. In the first round, which already showed a high mutual agreement between the annotators, the annotators (terminologists specialized in technical documentation) act independently; the agreement between the annotators, the annotators (terminologists involved in the work otherwise. The clustering procedure consisted of two rounds. In the first round, which already showed a high mutual agreement between the annotators, the annotators (terminologists specialized in technical documentation) act independently; the second round was a consensus round. Then, the verbs have been clustered using the clustering strategy that performed best in our trials: the optimized repeated bisections strategy. As already pointed out before, the best results were achieved for 60 clusters (i.e., k = 60).

To evaluate the performance of automatic clustering, we compute the purity of the obtained clusters. Cluster purity is a simple and transparent evaluation measure of clustering quality [35]:

\[
\text{purity}(C, \Omega) = \frac{1}{N} \sum_{i=1}^{k} \max_{j} \left| w_i \cap c_j \right|
\]

where \(\Omega = \{w_1, w_2, \ldots, w_k\}\) is the set of automatically derived clusters, \(C = \{c_1, c_2, \ldots, c_n\}\) is the set of gold standard clusters, and \(N\) is the total number of elements in the whole collection (321 in our experiment).

To compute purity, each cluster \(w_i\) is assigned a gold standard cluster \(c_j\) with which it has the maximal overlap, such that the purity is reflected by the number of elements of \(w_i\) correctly assigned to \(c_j\), normalized by \(N\). Bad clustering has a purity value close to 0, while perfect clustering has a purity value of 1. In our experiment, we

---

10 As already pointed out in Sections 1 and 2, the vast majority of the works focus on a restricted set of relations.
achieved a cluster purity of 0.732; a baseline that takes into account only the first sense of each verb in WN (but otherwise performs under the same parameter settings) achieves a purity of 0.531. Korhonen et al. [31] achieved a similar purity (namely 0.72) when clustering 192 verbs from the biomedical domain with the Information Bottleneck [53] and Information Distortion [17] methods into 53 clusters (against a gold standard divided into 50 classes).

5.3. Evaluation of cluster labeling

The evaluation of cluster labeling was carried out by human evaluators. For the evaluation, we use 60 verb clusters (see SubSection 4.2) as the list of clusters to name. The 60 clusters were presented to three judges, together with the labels assigned to each of the clusters by our system and by a human expert (the gold standard labels), such that the judges did not know the source of a label. For each cluster, the judges were asked to qualify both labels as 'correct', 'partially correct' or 'incorrect'; see Table 8 for the evaluation results.

Table 8 suggests that the Freq strategy, which chooses as the label of a cluster its most frequent member, shows better results, achieving a 78% of correctness. The VHyper strategy implements a common approach to lexical cluster labeling (see Section 2.3) and can thus be considered as an initial baseline. It shows significantly worse results than Freq, achieving only 43% of accuracy. This is, as has already been seen in Section 4.3, largely because the hyperonyms in WN tend to be too abstract to serve as a label of their hyponyms, as was, for example, also the case with move for the cluster [disperse, propagate].

A second baseline strategy that arbitrarily chooses a member of a given cluster as the label of this cluster achieves a 31% match with the gold standard labels. To interpret this figure, it is important to know that the gold standard labels chosen by the human

<table>
<thead>
<tr>
<th>Table 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering labeling evaluation with human judges.</td>
</tr>
<tr>
<td>Gold standard</td>
</tr>
<tr>
<td>Freq</td>
</tr>
<tr>
<td>VHyper</td>
</tr>
<tr>
<td>Baseline</td>
</tr>
</tbody>
</table>

Table 9

<table>
<thead>
<tr>
<th>Patent corpus IDs</th>
</tr>
</thead>
</table>
expert often coincide with the members in the clusters. Obviously, the result of the baseline depends on the cluster size.

6. Discussion

Let us examine the results of the evaluation of the three stages of our methodology.

The main reason of failure in our procedure of verbal relation extraction was an erroneous MiniPar output structure, which meant that no valid DSynt-structure could be derived. The failures in Cimiano et al.’s chunk-based procedure (see also Section 2) were due to the fact that:

(i) the two patterns they use do not capture all verbal relations (recall that they use \(NP1, V, NP2\) and \(NP1, V, PP, NP2\) to search for the argumentative structure of a verbal relation \(V\));
(ii) \(NP1, NP2\), and \(PP\) are not necessarily arguments of \(V\);
(iii) the relevant \(NP1, NP2\) or \(PP\) are situated at a long distance from \(V\) and are thus not captured by the patterns.

As indicated, our clustering strategy achieves a similar performance as [31], which is one of the very few works that addresses the problem of clustering of verbs from a specialized domain. A re-implementation of [31] with the purpose to apply it to our data for a direct comparison seems unfeasible, as this would be based on extensive previous work of the first author on the acquisition and clustering of subcategorization frames [30,32]. However, the similarity of the settings of both works suggests that such a re-implementation is not even necessary. On the other hand, as already argued by Korhonen et al. [31], domain-specific and general discourse lexical classifications cannot be easily compared due to major differences between the goals of the two types of tasks.

The measure of cluster purity used to evaluate the performance of the clustering strategy can be also used to verify our assumption that the one-sense-per-patent-domain assumption and to cross-check the similarity of all WN-senses of a verb \(V\) with all WN-senses of the verb \(V\). This permits us to dispense with the costly task of word sense disambiguation (WSD). For this purpose, we selected 100 verbs from 27 gold standard clusters of our verbal relation set and disambiguated them manually. The disambiguated verbs were clustered, again manually, according to the similarity of their senses in WN. The purity of the obtained clusters against the original 27 clusters ranged at 0.64. In a second round, the 100 verbs were again clustered, but this time automatically by our strategy described in Section 4.2 without prior disambiguation. The cluster purity was 0.71. From this outcome, we deduce that our working assumption that we can dispense with a WSD stage without major harm for the results of clustering is valid. Korhonen et al. [31] came to analogous conclusion concerning uniformity of verb senses in the biomedical domain when examining the purity of the clusters there.

According to the qualitative evaluation, the performance of one of the cluster labeling strategies, ‘Freq’, is similar to that of a human, while the strategy VHyper, which is most commonly used for lexical labeling, performs significantly poorer. This is partly due to the fact that most of the works on lexical labeling target the labeling of nominal rather than verbal clusters and WordNet, which is used as reference resource, has, in general, very flat verbal hierarchies.

7. Conclusions and future work

We have presented a three-step verbal content relation extraction strategy that, unlike most of the approaches in the literature, is not restricted to a limited set of relations. To be able to detect all verbal content relations in patent claim corpora, our strategy operates with deep-syntactic dependency structures rather than with lexical or surface-syntactic patterns. The linguistic constructions in patent claims are extremely complex. The fact that our strategy is able to cope with them with a rather good performance lets us assume that it can be also extrapolated to other genres. We also have evidence that it performs similarly well for relations rendered by predicative nouns and adjectives.

Still, we believe that our strategy can be further improved and are working on this. One option is to use a more powerful parser. Currently, we carry out experiments with Bohnet [4]’s stochastic parser, which is about to be trained on a SSysTs-annotated corpus. This parser demonstrated a very good performance and robustness in the CoNLL ’09 competition, and we are confident that it will also perform very well in our application. We also plan to experiment with cluster labeling that searches for labels by contrasting clusters among each other and with more refined clustering parameters that are based on linguistic criteria.

Acknowledgments

The work described in this paper has been partially carried out in the scope of the PATExpert project, funded by the European Commission under the contract number FP6-ICT-028116. Many thanks to all colleagues in PATExpert and, in particular, to our colleagues from the NLP group at UF, Nadjeet Bouayad-Agha, Gerard Casamayor and Simon Mille, for their valuable contributions and help. The first draft of the paper was read and commented upon by Nadjeet Bouayad-Agha, Rogelio Nazar, and Horacio Saggion. Josh Holden corrected our English. We would like to express our heartfelt gratitude to all of them. We would also like to thank the three anonymous reviewers of the Knowledge-Based Systems Journal for their very helpful and insightful comments on the previous versions of the paper.

Appendix A. References to the patents used in the experiments

See Table 9.

References


11 A considerable number of different parsers both statistical and rule-based are available to date; see, e.g., [14,29,39,42,54,27].