Implementing WordNet Measures of Lexical Semantic Similarity in a Fuzzy Logic Programming System*

PASCUAL JULIÁN-IRANZO
Dept. of Information Technologies and Systems, University of Castilla-La Mancha, 13071 Ciudad Real, Spain
(e-mail: Pascual.Julian@uclm.es)

FERNANDO SÁENZ-PÉREZ
Faculty of Computer Science, Complutense University of Madrid, 28040 Madrid, Spain
(e-mail: fernan@sip.ucm.es)

submitted 25 November 2019; revised 25 November 2020; accepted -- --

Abstract

This paper introduces techniques to integrate WordNet into a Fuzzy Logic Programming system. Since WordNet relates words but does not give graded information on the relation between them, we have implemented standard similarity measures and new directives allowing the proximity equations linking two words to be generated with an approximation degree. Proximity equations are the key syntactic structures which, in addition to a weak unification algorithm, make a flexible query-answering process possible in this kind of programming language. This addition widens the scope of Fuzzy Logic Programming, allowing certain forms of lexical reasoning, and reinforcing Natural Language Processing applications.

KEYWORDS: Fuzzy Logic Programming, WordNet, Proximity Equations, System Implementation

1 Introduction and Motivation

Fuzzy Logic Programming (Lee 1972) integrates concepts coming from fuzzy logic (Zadeh 1965) into logic programming (van Emden and Kowalski 1976) in order to deal with the essential vagueness of some problems by using declarative techniques. In recent years there has been renewed interest in this field, involving multiple lines of work. When the fuzzy unification algorithm is weakened using a similarity relation (i.e., a reflexive, symmetric, transitive, fuzzy binary relation) the approach is usually called Similarity-based Logic Programming (Fontana and Formato 1999, Fontana and Formato 2002, Loia et al. 2001, Sessa 2002).

We have extended Similarity-based Logic Programming by introducing new theoretical concepts and developing two fuzzy logic programming systems: Bousi~Prolog (BPL for short)
(Rubio-Manzano and Julián-Iranzo 2014; Julián-Iranzo and Rubio-Manzano 2017) and FuzzyDES (Julián-Iranzo and Sáenz-Pérez 2017; Julián-Iranzo and Sáenz-Pérez 2018b). Their syntax is based on the clausal form, and they embody a Weak SLD (WSLD) resolution operational semantics, which uses a fuzzy unification algorithm based on the concept of proximity relation (i.e., a fuzzy binary relation supporting unification that, although reflexive and symmetric, is not necessarily transitive) (Julián-Iranzo and Rubio-Manzano 2015; Julián-Iranzo and Sáenz-Pérez 2018a). A proximity relation is defined by proximity equations, denoted by $a \sim b = \alpha$, whose intuitive reading is that two constants (either $n$-ary function symbols or $n$-ary predicate symbols), $a$ and $b$, are approximate or similar with a certain degree $\alpha$.

For instance, assume a deductive database that stores information about people and their family relationships encoded using the Bousí~Prolog language (see Fig. 1). In a Prolog system (without proximity equations), asking about the progenitors of isaac with the query \texttt{progenitor(X,isaac)} produces no answer. However, Bousí~Prolog answers $X=abraham$ with 0.9 and $X=sara$ with 0.9 thanks to its proximity-based unification algorithm. Since we have specified that progenitor is close to ancestor with degree 0.9, these two terms can “weakly” unify with approximation degree 0.9, leading to a refutation.

Here, the proximity equations are axiomatically given by the programmer. It would be interesting if the system could provide assistance through its connection to a lexical resource such as WordNet ([Fellbaum 1998]; [Fellbaum 2006]; [Miller 1995]). This study, therefore, deals with the generation of the set of proximity equations both automatically and with a minimal intervention by the programmer. However, the motivation for integrating WordNet into our logic systems goes beyond this simple help function. We provide the Prolog implementation, \texttt{wn\_connect}, to connect to WordNet with a number of similarity measures and convenient predicates to be used either in isolation or integrated into Bousí~Prolog, allowing reasoning with linguistic terms. Unlike Distributional Semantic Models such as Word Embeddings or other statistical approaches, WordNet-based techniques do not require training and facilitate explainability (Santus et al. 2018).

The usefulness of this proposal lies in its inclusion of applications such as text mining in information retrieval, text classification, and even sentiment analysis ([Allahyari et al. 2017]; [Baeza-Yates and Ribeiro-Neto 2011]; [Serrano-Guerrero et al. 2015]) (see also 6.2).
Implementing WordNet Lexical Similarity Measures

2 The Lexical Resource WordNet and Prolog

WordNet is a lexical English language database. Words of the same syntactic category are grouped into sets of synonyms called *synsets*. Roughly speaking, the words of a synset have the same meaning in a specific context and they represent a *concept* (or word sense). Each synset has a *synset ID*. Because a word has different senses (meanings), it can belong to different synsets. WordNet is structured as a semantic net where words are interlinked by lexical relations, and synsets by semantic relations. Synonymy and antonymy are the major lexical relations. Semantic relations serve to build knowledge structures (i.e., networks of synsets –concepts–). Nouns, as well as verbs, are interconnected by the *hyponymy* relation (IS-A relation), which links specific concepts to more general ones. Hypernymy is the opposite relation, that is, a hypernym is a word whose meaning includes a group of other words. Both relations are transitive. Note also that both nouns and verbs are organized as separate hierarchical structures.

WordNet can be accessed either via a web interface or locally. There exists a WordNet 3.0 database version released by Eric Kafe which can be found at: https://github.com/ekaf/wordnet-prolog. The information stored in WordNet is provided as a collection of Prolog files. Each file contains the definition of what is called an *operator*, corresponding to a WordNet relation. Files are named wn_<operator>_pl, where <operator> is the name of a specific operation (relation). Therefore, each WordNet relation is represented by a Prolog predicate which is stored in a separate file and defined by a set of Prolog facts. The specifications of these predicates are detailed in (Fellbaum and et al. 2006). We now describe the predicates of greatest interest for this study.

The file wn_s_pl contains all the information about words stored in WordNet. It defines the *s* operator, which has an entry for each word. The structure of the *s* operator is *s*(synset_id, w_num, word, ss_type, sense_number, tag_count), where w_num, if given, indicates which word in the synset is being referred to. The words in a synset are numbered serially, starting with 1. The third argument is the word itself (which is represented by a Prolog atom). The ss_type parameter is a one character code indicating the synset type: n (noun); v (verb); a (adjective); s (satellite adjective) and r (adverb). The sense_number parameter specifies the sense of the word, within the part of speech encoded in the synset_id. The higher the sense number, the less common the word. Finally, the tag_count indicates the number of times the word sense was found in the sense-tagged text corpus of Semantic Concordances (Miller et al. 1993), which was generated from the Brown Corpus (Francis and Kucera 1979), using WordNet as a lexicon. The Brown Corpus was inspected word by word, including sense-tags for each one. A higher tag count number means that the word is more common than others with a lower tag count. In Section 3.4 we illustrate the meaning of some of these parameters through an example.

The file wn_hyp_pl stores hypernymy relations in the binary predicate hyp(synset_ID1, synset_ID2) specifying that the second synset is a hypernym of the first synset. This semantic relation only holds for nouns and verbs. Because hyponymy is the inverse relation of hypernymy, the operator hyp also specifies that the first synset is a hyponym of the second synset.

3 WordNet and Lexical Semantic Similarity

WordNet relates words but does not give their degree of relationship. Measuring lexical semantic similarity has many applications for Natural Language Processing (NLP), and its integration into a fuzzy logic programming system such as Bousi~Prolog is appropriate because of
its proximity-based operational semantics. The syntax of our language uses symbols (words) that are endowed with a fuzzy semantics via proximity equations. We are therefore interested in techniques for measuring the similarity degree between words to facilitate the construction of proximity equations with linguistic criteria. Semantic similarity quantifies how alike two words are (more precisely: how similar the concepts they denote are).

Similarity measures are limited to noun pairs and verb pairs because WordNet organizes nouns and verbs into hyponymy/hypernymy-based hierarchies of concepts (synsets).

Although a large number of measures of semantic relatedness and similarity have been proposed, they are only implemented by a limited number of tools. WordNet::Similarity (Pedersen et al. 2004) is perhaps the most prominent. This tool has three similarity measures based on counting edges between concepts (PATH, WUP (Wu and Palmer 1994) and LCH (Leacock and Chodorow 1998)), and another three based on information content (RES (Resnik 1995), JCN (Jiang and Conrath 1997) and LIN (Lin 1998)).

Table 1 summarizes some features of these measures, with the measure name in the first column, its type (either counting Edges Based –EB– or Information Content –IC–) in the second, its description in the third, and its range in the last. In order to understand the description of similarity measures accurately, we introduce the following standard definitions and notations used when working in the framework of WordNet:

- We differentiate between “words” and “concepts”. We use the term “word” as shorthand for “word form,” and the term “concept” (i.e., “synset”) to refer to a specific sense or word meaning. Words will be denoted by the letter \( w \), and concepts by the letter \( c \), possibly with subscripts. A concept can also be seen as a word \( w \) of type \( t \) with sense \( s \) and denoted by \( w: t:s \) (which we often call word term or pattern).

- Similarity measures use so-called HyperTrees (Hyponym Trees). These are IS-A hierarchies which are a consequence of the hyponymy relation between concepts. Despite their name, HyperTrees are not really trees because a concept can be linked to a hyponym concept through different paths. Moreover, in practice, the branches of a HyperTree are manipulated independently as hypernym chains. Given a HyperTree, the length of the shortest path from synset \( c_1 \) to synset \( c_2 \) is denoted by \( \text{len}(c_1, c_2) \). The depth of a node \( c \) is the length of the shortest path from the global root to \( c \), i.e., \( \text{depth}(c) = \text{len}(\text{root}, c) \). The “global root” is a virtual root that we introduce into the IS-A hierarchy of either nouns or verbs for technical reasons.

- The least common subsumer (LCS) of two concepts \( c_1 \) and \( c_2 \) is the most specific concept they share as an ancestor. It is denoted by \( \text{lcs}(c_1, c_2) \). An example illustrating the notion of LCS is shown in Section 5.2.

RES, JCN and LIN measures are based on the notion of Information Content (IC) (Resnik 1995). For a concept \( c \), \( \text{IC}(c) = -\ln(p(c)) \), where \( p \) is the probability of finding an instance of the concept \( c \) in a corpus. In our case, this probability is measured in terms of a relative frequency of use (or frequency count) of the concept \( c \) stored in WordNet, which is a measure of the number

---

1 Note that “lexical semantic relatedness” is a broad concept that subsumes “lexical semantic similarity”. There are many different forms in which two words can be related without being similar: for instance, “car” and “petrol” are closely related, but they are not similar. From a pragmatic point of view, and to distinguish one type of measure from another, it is usual to reserve the name “relatedness” for those that measure features other than similarity.
Table 1. Some similarity measures and their features

| Measure | Type | Description                                      | Range  |
|---------|------|--------------------------------------------------|--------|
| PATH    | EB.  | \(\text{sim}_{\text{PATH}}(c_1, c_2) = 1/\text{len}(c_1, c_2)\) | [0,1]  |
| WUP     | EB.  | \(\text{sim}_{\text{WUP}}(c_1, c_2) = 2 \times \text{depth}(lcs(c_1, c_2)) / \text{Depth}(c_1) \times \text{Depth}(c_2)\) | [0,1]  |
| LCH     | EB.  | \(\text{sim}_{\text{LCH}}(c_1, c_2) = -\log\left(\frac{\text{len}(c_1, c_2)}{\max\{\text{depth}(c_1), \text{depth}(c_2)\}}\right)\) | [0,\infty[ |
| RES     | IC   | \(\text{sim}_{\text{RES}}(c_1, c_2) = \text{IC}(\text{lcs}(c_1, c_2))\) | [0,\infty[ |
| JCN     | IC   | \(\text{sim}_{\text{JCN}}(c_1, c_2) = \frac{1}{\text{IC}(c_1) + \text{IC}(c_2) - 2 \times \text{IC}(\text{lcs}(c_1, c_2))}\) | [0,\infty[ |
| LIN     | IC   | \(\text{sim}_{\text{LIN}}(c_1, c_2) = \frac{2 \times \text{IC}(\text{lcs}(c_1, c_2))}{\text{IC}(c_1) + \text{IC}(c_2) + \text{IC}(\text{lcs}(c_1, c_2))}\) | [0,1]  |

of times that it occurs in the corpus of Semantic Concordances. Specifically,

\[ p(c) = \text{Frequency}(c) / \text{Frequency}(\text{Root}) \]

where \(\text{Frequency}(c)\) is computed by adding the \text{Tag.count} of the concepts subsumed by the concept \(c\) and \(\text{Root}\) is the concept (virtual or not) on the top of the concept hierarchy.

4 Formal Setting

This section recalls and extends some formal background of BPL and its relation to lexical similarity. Given a universe \(U\), proximity equations extensionally define a binary fuzzy relation \(\mathcal{R} : U \times U \rightarrow [0,1]\). A \(\lambda\)-cut is a user-defined threshold such that \(\mathcal{R}_\lambda = \{ (x, y) \mid \mathcal{R}(x, y) \geq \lambda \}\). A fuzzy relation can have some properties attached, for any \(e, e_1, e_2, e_3 \in U\): Reflexive (\(\mathcal{R}(e, e) = 1\)), Symmetric (\(\mathcal{R}(e_1, e_2) = \mathcal{R}(e_2, e_1)\)), and \(\triangle\)-Transitive (\(\mathcal{R}(e_1, e_3) \geq \mathcal{R}(e_1, e_2) \triangle \mathcal{R}(e_2, e_3)\)), where the operator \(\triangle\) is an arbitrary t-norm. A fuzzy relation with the reflexive and symmetric properties is a proximity relation. If in addition it has the \(\triangle\)-transitive property, it is a similarity relation.\(^2\)

A weak unification of terms builds upon the notion of weak unifier of level \(\lambda\) for two expressions \(\delta_1\) and \(\delta_2\) with respect to \(\mathcal{R}\) (or \(\lambda\)-unifier): a substitution \(\theta\) such that \(\mathcal{R}(\delta_1 \theta, \delta_2 \theta) \geq \lambda\), which is the unification degree of \(\delta_1\) and \(\delta_2\) with respect to \(\theta\) and \(\mathcal{R}\). There are several weak unification algorithms (Julían-Iranzo and Sáenz-Pérez 2018a) based on this notion and on the proximity-based unification relation \(\Rightarrow\), which defines a transition system (based on (Martelli and Montanari 1982)). This relation, applied to a set of unification problems \(\{\delta_i \approx \delta'_i | i \leq n\}\), can yield either a successful or a failed sequence of transition steps. In the first case, both a successful substitution and a unification degree are obtained (detailed in, e.g., (Julían-Iranzo and Sáenz-Pérez 2018a)). The notion of weak most general unifier (wmgu) \(\theta\) between two expressions, denoted by \(\text{wmgu}_\lambda^\delta(\delta_1, \delta_2)\), is defined as a \(\lambda\)-unifier of \(\delta_1\) and \(\delta_2\) such that there is no other \(\lambda\)-unifier which is more general than \(\theta\). Unlike in the classical case, the wmgu is not unique. However, our weak unification algorithm computes a representative wmgu with approximation degree greater than or equal to any other wmgu.

Given a fuzzy logic program \(\Pi\) with rules \(\langle A \leftarrow Q; \delta \rangle\), where \(A\) is an atomic formula, \(Q\) is either empty or a conjunction of \(n \geq 0\) atomic formulas \(B_i\), and \(\delta\) is the degree of the rule, an operational semantics can be defined as a transition system with a transition relation \(\Rightarrow_{\text{wsld}}\) which, in particular, includes the (transition) rule:

\(^2\) Lexical semantic similarity and a similarity relation are two different concepts. The first simply provides a degree of similarity between words, but it is not a similarity relation in the sense defined above, with the reflexive, symmetric and \(\triangle\)-transitive properties. On the practical side, we use the WordNet similarity measures to obtain the approximation degree that we use when automatically constructing the proximity equations.
\[
(\langle \leftarrow A', Q \rangle, \theta, \alpha) \Rightarrow_{\text{wld}} (\langle \leftarrow Q, Q' \rangle, \theta, \sigma, \delta \Delta \beta \Delta \alpha)
\]

if \( \langle A \leftarrow Q ; \delta \rangle \in \Pi \), \( \sigma = \text{wmgu}_{A'}(A, A') \neq \text{fail} \), \( \mathcal{R}(A, A', \sigma) = \beta \geq \lambda \), and \( (\delta \Delta \beta \Delta \alpha) \geq \lambda \).

A fuzzy logic program \( \Pi \) is translated into a logic program by: linearizing heads, making the weak unification explicit, and explicitly computing the approximation degree. Essentially, given a graded rule \( \langle p(t_n) \leftarrow Q ; \delta \rangle \), for each \( \mathcal{R}(p, q) = \alpha \in \Pi \) with \( \alpha \geq \lambda \), generate the clause:

\[
q(t_n) \leftarrow (\delta \Delta \alpha) \land t_1 \land \ldots \land t_n \approx t_n \land Q
\]

where \( \approx \) is the weak unification operator, \( t_i \) are terms, \( x_i \) are variables, and \( \delta \Delta \alpha \) abbreviates the goal \( \delta \Delta \alpha \geq \lambda \).

We assume 3-arity predicates for lexical similarity measures with the pattern \( \langle w_1 : t_1 : s_1, w_2 : t_2 : s_2, d \rangle \), where \( w_1, t_1, s_1 \) are word terms and \( d \in (0, 1] \) is a normalized semantic similarity degree. Any element \( e \) in the semantics of a lexical similarity measure \( m \) can be used to generate a proximity equation \( \mathcal{R}(w_1, w_2) = d \) defining \( \mathcal{R} \). Depending on the fuzzy relation we decide to work with, Bousi~Prolog generates several types of closure starting from the proximity equations defining \( \mathcal{R} \). Specifically, since a similarity relation requires all of the three properties (in particular, transitivity), its intension is its reflexive, symmetric, \( \Delta \)-transitive closure. This allows for both manual and automatic generation of proximity equations relating similar words, including words that are not directly related by \( m \) (cf. Section 5).

Section 5 will show how to integrate WordNet and the aforementioned lexical semantic similarity measures into the state-of-the-art fuzzy logic programming system Bousi~Prolog.

### 5 Integrating WordNet into Bousi~Prolog

Bousi~Prolog comprises three subsystems with a total of nine modules. The \( \text{wn-connect} \) sub-system provides the basis for the connection between WordNet and the Bousi~Prolog system. \( \text{wn-connect} \) is a software application in itself with ten Prolog modules, which implements predicates for managing synsets, hypernyms and hyponyms, giving support to the \( \text{wn_sim_measures} \) and \( \text{wn_ic_measures} \) modules which, in addition, implement the standard similarity measures defined in Section 4. We now offer a summary of the base modules:

- The \( \text{wn_synsets} \) module implements predicates to retrieve information about words and synsets stored in WordNet. It uses the \( \text{wn} \) module implemented by Jan Wielemaker which exploits SWI-Prolog demand-loading and Quick Load Files (QLF) for ‘just-in-time’ fast loading.
- The \( \text{wn_hypernyms} \) module implements predicates to retrieve information about hypernyms of a concept (synset). It uses the modules \( \text{wn_synsets} \) and \( \text{wn_utilities} \). Notably, the predicate \( \text{wn_hypernyms}/2 \) returns a list \( \text{List_SynSet_HyperNym} \) of hypernyms (as synset identifiers) for a word term Hypernym.
- The \( \text{wn_hyponyms} \) module implements predicates to retrieve information about hyponyms of a concept (synset). Remarkably, the predicate \( \text{wn_gen_all_hyponyms_of}/2 \) generates all the hyponyms of a concept (Synset_ID), and is especially useful for computing the information content of a concept.

---

3 [https://dectau.uclm.es/bousi-prolog](https://dectau.uclm.es/bousi-prolog)
4 [https://dectau.uclm.es/bousi-prolog/2018/08/27/applications/](https://dectau.uclm.es/bousi-prolog/2018/08/27/applications/) supplies the source files with the code and detailed comments of its implementation.
5 [https://github.com/JanWielemaker/wordnet](https://github.com/JanWielemaker/wordnet)
5.1 Implementing Similarity Measures

A first step for a more ambitious goal is to automatically extract semantic similarity information from WordNet IS-A hierarchies, and other attributes as the frequency of use as explained before in Section 3. Here, we describe in broad strokes the implementation of similarity measures based on edge-counting (module `wn_sim_measures`) and some insights about those based on information content (module `wn_ic_measures`).

To a greater or lesser extent, all edge-counting similarity measures are based on the computation of the LCS of two words (more accurately, concepts). The predicate `wn_sim_measures: lcs/6` returns the LCS of two words `Word1` and `Word2`, and also measures depth in their respective HyperTrees. Roughly speaking, it computes the HyperTrees of `Word1` and `Word2` and compares them from their roots, returning the `synset_ID` previous to the first mismatch (which is the LCS of both concepts). Additionally, the predicate `lcs/6` returns the depths for LCS, `Word1` and `Word2` for reasons of efficiency: we want to go through the hypernym lists only once, so these quantities are calculated when computing the LCS.

The computation of the hypernyms of a concept is carried out by the predicate `wn_hypernyms: hypernym_chain/2`, which computes a list (SynSet_HyperNyms) of `synset_IDs` designating the hypernyms of a concept (SynSet_Hypernym). It thus computes a HyperTree that will be used in the former comparison to compute the LCS.

Once the depths of the LCS and the words to be compared are known, it is easy to compute the relationship degree between them by following the guidelines given in Section 3. For example, the WUP measure is implemented by the predicate `wn_wup/3`, which takes two concepts (expressed as word terms of the form `Word:SS_type:Sense_num`) and returns the degree of similarity between them. It relies on the private predicate `wup/3` that calls `lcs/6` to generate and inspect a pair of HyperTrees associated to `Word1` and `Word2`, and obtains the similarity degree between both words (according to that pair of HyperTrees). Because a concept can have more than one HyperTree, several pairs of HyperTrees are possibly considered, and a list of similarity degrees is obtained for each of these pairs of HyperTrees. Finally, the maximum degree in the list is selected as a result.

Regarding similarity measures based on the information content, the key idea lies in the implementation of the notion of frequency of use. The operator `wn_s/6` stores information on how common a word is. The `tag_number` indicates the number of times the word was found in a text corpus: the higher the number, the more common the word is. This parameter can therefore be employed to obtain the use of a word and, summing the `tag_number` of all words in a synset, the specific use associated to a whole synset (i.e., to a concept) can be obtained. Then, the frequency of use of a concept is obtained by adding the “synset tag num” of all concepts subsumed by that concept.

As explained in Section 3, the information content of a concept is a function of the ratio between the frequency of use of that concept and the frequency of use of the root concept of the hierarchy. Finally, the information-content-based measures are computed as shown in Table I for specific predicates. Note that we have taken the option of smoothing the frequencies of use with a value of 0, which we substitute for a very small number close to 0. So, some relationship values do not exactly match those that would be obtained when using tools like `wordnet::similarity` (Pedersen et al. 2004).

Finally, 6.1 includes a performance comparison between the similarity measures implemented for Bousi~Prolog and other systems.
5.2 Directives for Generating Proximity Equations

Bousi–Prolog can load both ontologies (consisting of proximity equations) and fuzzy logic programs (with fuzzy logic rules and possibly proximity equations). Thus, it would be of interest to use the similarity measures implemented in the last section to automatically construct such ontologies.

In order to define the semantic similarity between selected concepts, we provide a Bousi–Prolog directive for automatically generating the proximity equations which define an ontology:

• :-wn_gen_prox_equations(+Msr, +LL_of_Pats)

where Msr is the similarity measure which can be any of: path, wup, lch, res, jcn and lin. The second argument LL_of_Pats is a list for which each element is another list containing the patterns that must be related by proximity equations. The pattern can be either a word or the structure Word:Type:Sense, where Word is the word, Type is its type (either n for noun or v for verb), and Sense is the sense number in its synset. Note that, because similarity measures only relate nouns with nouns, and verbs with verbs, the words of a set must be of the same part of speech. If the pattern is simply a word, then a sense number of 1 is assumed, and its type is made to match all other words in the same list.

An example of this directive is:

:-wn_gen_prox_equations(wup, [[man,human,person], [grain:n:8, wheat:n:2]]).

In this case, as only words are provided in the first list, the sense number is 1, and their types are equal by pairs (nouns for these words). The second list explicitly specifies the pattern of each word to be related. Then, excluding, for reasons of simplicity, reflexive and symmetric entries, the following proximity equations are generated for a lambda cut of 0:

sim(man, human, 1, 0.56).
sim(man, person, 1, 0.8888888888888888).
sim(person, man, 1, 0.8888888888888888).
sim(human, person, 1, 0.6086956521739131).
sim(grain, wheat, 0, 0.2608695652173913).

Note that there are two blocks, numbered with 1 for the first four equations, and with 0 for the last one. Clearly, words in the first list are not made to be related to those in the second list, and therefore they must occur in different blocks. In addition, proximity equations are generated only for the words stored in WordNet.

Another form of this directive automatically builds an ontology in terms of the tokens in the BPL program by including auto in its second argument. Only the symbols that occur in a program are related, because it would not be practical to relate the symbols of the program with all those that occur in WordNet.

5.3 Implementing the Generation of Proximity Equations

Bousi–Prolog processes a file (either a program or an ontology) with the load command ld file of the BPL Shell module (named bplShell), where its argument is the name of the file to load (with default extension bpl). Upon execution of this command, a source file (file.bpl) is parsed, compiled to Prolog (file.tpl), and consulted.

When parsing a directive :-wn_gen_prox_equations, it is first checked for validity, and then
replaced in the target Prolog file with the proximity equations corresponding to the pairs formed with the symbols derived from its arguments. As explained, there are two cases for this directive, and they are handled in a different way:

- **Explicit indication of words to be related.**
  Here, the proximity equations can be directly generated from each list of words, kept in the memory (as asserted Prolog facts) and outputted to the translated program in the .tpl file at a later stage. The procedure is as would be expected: for each pair of different words \( W_1 \) and \( W_2 \) in a list, generate the proximity equation \( \text{sim}(W_1, W_2, D) \), where \( D \) is the approximation degree for the normalized measure given as the first parameter of the directive. Normalization is required because measures are generally not on the interval \((0, 1]\) which is the range for proximity equations.

- **Automatic generation of proximity equations.**
  This case is different from the former because, when processing the directive, the rules in the program have not yet been parsed, so their tokens are not available. It is therefore processed after parsing the remaining program, by performing a syntactic analysis in order to extract the sets of constant, functor and predicate identifiers and adding the resulting proximity equations for each separate set of tokens (with the same shape as in the other case) to the memory.

The directives that generate proximity equations are based on the private predicate \texttt{gen_prox_equation}. It generates a proximity equation \( \text{sim}(\text{Word}_1, \text{Word}_2, \text{NormalizedDegree}) \) in terms of a given measure (\texttt{Measure}) and a pair of words, which can be completely specified with either a pattern or only with its syntactic form as plain words. In this last case, their first sense number is selected and the same word type is enforced.

### 5.4 Accessing WordNet

The \texttt{wn_connect} subsystem must be made visible before using the built-ins (public predicates) defined in its modules. In Bousi-~Prolog, WordNet and a wide repertoire of built-in predicates which are implemented by the \texttt{wn_connect} modules can be accessed either by the directive \texttt{:-wn_connect} in a program or interactively with \texttt{ensure_loaded(wn(wn_connect))} at the command prompt.

Nearly all the predicates implemented in the \texttt{wn-connect} subsystem are crisp, returning the top approximation degree. For instance, the predicate \texttt{wn_word_info/1} merges the information provided by the predicate \texttt{wn_s/6} (which stores information about a synset) and \texttt{wn_g/2} (which contains an explanation/definition of the concept represented by the synset and example sentences). Figure 2 shows the first answer to the query \texttt{wn_word_info(cat)}. This is telling us that the first sense (Sense number = 1) of the word form “cat” in the part of speech of nouns (Synset type = \texttt{n} —i.e., a noun—) refers to the concept: “feline mammal usually having thick soft fur and no ability to roar etc.”. There are six more answers for noun-related senses and two more for verb-related senses.

However, the binary similarity predicates (\texttt{wn_path/2}, \texttt{wn_wup/2}, \texttt{wn_lch/2}, etc.) are fuzzy predicates that return the similarity degree of two concepts. We also maintain ternary predicates available to programmers, since they provide direct access to the approximation degree \( D \), which can be very useful for its explicit handling. Thanks to the repertoire of built-in predicates implemented in the \texttt{wn-connect} subsystem, the user of the BPL system can extract information from
?- wn_word_info(cat).
INFORMATION ABOUT THE WORD 'cat':
  Synset_id = 102121620
  Word Order num. = 1
  Synset type (n:NOUN, v:VERB, a:ADJ., s:ADJ. SAT., r:ADV.) = n
  Sense number = 1
  Tag_count = 18
  ---------------
  Gloss:
feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats
  ---------------
  true

Fig. 2. A query for obtaining all relevant information about a word (e.g. the word “cat”)

Fig. 3. Hypernyms of the word god (all senses)

WordNet, deepening into the structure of the relationships between its linguistic terms. This becomes especially evident for the predicate `wn_display_graph_hypernyms/1`. Figure 3 shows its outcome for the hypernym hierarchy of all senses of the word god.

Moreover, with these built-in predicates, a certain form of linguistic reasoning is possible. For example, the predicate `wn_lcs/2`, which computes the LCS of a set of concepts, can help to obtain the most specific generalization of a set of concepts and to contribute to knowledge discovery.

---

6 In Figure 3, each node draws the representative word of the respective synset (i.e., those with `wn.num` equal to one). This figure also illustrates how a concept can be linked to a hypernym concept through different paths in (a subset of) the WordNet IS-A hierarchy.
Implementing WordNet Lexical Similarity Measures

?- wn_lcs([lion, leopard, cougar, cat], LCS_SS_ID),
    wn_synset_components(LCS_SS_ID, Words_LCS_SS_ID).
LCS_SS_ID = 102120997,
Words_LCS_SS_ID = [feline:n:1, felid:n:1].

Fig. 4. A query for obtaining the LCS or most specific generalization of a set of concepts.

Although in a database there only exists direct information about, e.g., lion, leopard, cougar and cat, it is possible to discover that this information is also pertinent for feline by using the predicate \texttt{wn\_lcs/2}. In Figure 4, the concept referred to by the \texttt{synset\_ID} 102120997 (grouping \texttt{[feline:n:1, felid:n:1]}) is the most specific concept of (the synsets of) lion, leopard, cougar and cat, that they share as a common ancestor in the IS-A hierarchy of WordNet. Furthermore, the predicate \texttt{wn\_gen\_hyponyms\_upto\_level/3}, which generates all the hyponyms of a concept (\texttt{Synset\_ID}) up to a certain depth level (\texttt{Level}), can also be used to generate an ontology of closely related terms to the given concept that can be used to implement flexible queries and text mining tasks. In particular, Fig. 4 illustrates an application of this work to text classification, also including some performance measures.

6 Experimental Assessment

In the following two subsections we perform experiments to find the performance of the implemented similarity measures and the cost of integrating WordNet into Bousi-Prolog. Instructions, programs and data to reproduce the experiments in these appendices have been made available at \url{https://dectau.uclm.es/bousi-prolog/wp-content/uploads/sites/3/2020/07/Published.zip}.

6.1 Evaluation of the Measures and Comparison with other Systems

In this section we evaluate the computational cost of the measures implemented in Subsection 5.1 comparing the results with other systems.

Specifically, we are using an implementation of WordNet::Similarity for Java (WS4J) developed by Hideki Shima. We use WS4J because it provides some time information that allows the cost of these measures to be appreciated. We are also using our own implementation of the WordNet-based similarity measures, but executed both by SWI-Prolog and Bousi-Prolog. This allows us not only to compare the performance of our measures integrated into Bousi-Prolog with WS4J, but also the overhead introduced by our implementation of Bousi-Prolog w.r.t. SWI-Prolog.

In the first experiment, we selected twelve words with the highest number of senses. Then we pair them obtaining six pairs of words. Afterwards, for each word in that pair we generate the Cartesian product of all their senses (\texttt{Word1:n\_Sense1, Word2:n\_Sense2}). Finally, we compute the similarity degree of these pairs, thus mimicking how WS4J operates\cite{WS4J}, measuring the overall time cost of the computation.

Table 2 shows the costs involved in the computation of the similarity degree of these six pairs.

---

\footnotetext[7]{WS4J is available at \url{https://github.com/Sciss/ws4j} and also has a web interface WS4J Demo at \url{http://ws4jdemo.appspot.com}.}

\footnotetext[8]{For WS4J, the parameter MFS (Most Frequent Sense) is set to false.}
of words for the three systems and the three edge-based measures. For each measure, ‘Time’ shows the elapsed time in milliseconds, ‘Lat.’ the latency in milliseconds/pair, and ‘Thr.’ the throughput in pairs/second. While BPL is at a small disadvantage with respect to SWIPL, WS4J is roughly four times as fast.

Table 2. Comparing Bousi~Prolog, SWI-Prolog and WS4J on six pairs of words with the highest number of senses

| System | PATH | WUP | LCH |
|--------|------|-----|-----|
|        | Time | Lat. | Thr. | Time | Lat. | Thr. | Time | Lat. | Thr. |
| BPL    | 966  | 0.02 | 44,572 | 959  | 0.02 | 45,400 | 959  | 0.02 | 44,943 |
| SWIPL  | 895  | 0.02 | 48,004 | 920  | 0.02 | 46,540 | 950  | 0.02 | 45,117 |
| WS4J   | 211  | 0.01 | 204,410 | 430  | 0.01 | 101,024 | 212  | 0.01 | 205,464 |

In a second experiment, we randomly generate pairs of noun and verb patterns (\texttt{Word1:Type:Sense1}, \texttt{Word2:Type:Sense2}) so Type is either n or v. Then, we generate the calls to a similarity measure, and finally, we measure the performance of Bousi~Prolog w.r.t. SWI-Prolog. Table 3 shows the results of this experiment. The numbers are the average after tree executions.

Table 3. Comparing Bousi~Prolog and SWI-Prolog on randomly generated pairs of patterns

| System | Edge-based | Information Content-based |
|--------|-------------|--------------------------|
|        | PATH | WUP | LCH | RES | JCN | LIN |
|        | (milliseconds/10,000 pairs) | (milliseconds/250 pairs) | | | | |
| BPL    | 1,438 | 1,403 | 1,597 | 34,680 | 36,221 | 34,982 |
| SWIPL  | 242  | 242  | 275  | 33,521 | 35,088 | 35,073 |

The analysis of the data in Table 3 reveals that for the BPL system the average Latency of edge-based measures is 0.14 ms/pair and the average Latency of IC-based measures is 141.17 ms/pair, while for SWI-Prolog they are 0.03 ms/pair and 138.24 ms/pair respectively. These results lead to an average ratio between both systems of 5.84 for edge-based measures and only 1.02 for IC-based measures, showing an acceptable overhead of Bousi~Prolog relative to SWI-Prolog for these tasks. In the first case, the overhead is more noticeable when traversing 10K word pairs than only 250 because tail recursion optimization is lost in the Bousi~Prolog to Prolog translation. Thus, optimizing this translation will be the subject of future work.

6.2 Applications to Text Classification and some Performance Results

Bousi~Prolog is well suited to making the query-answering process more flexible, due to its weak unification algorithm. In (Rubio-Manzano and Julián-Iranzo 2014) we discussed several practical applications where it can be useful, such as flexible deductive databases, knowledge-based systems, information retrieval, and approximate reasoning. Bousi~Prolog has been used in a number of real applications such as: text classification (or cataloging) (Romero et al. 2013).
knowledge discovery (Rubio-Manzano and Julián-Iránzo 2015), linguistic feedback in computer games (Rubio-Manzano and Triviño 2016), and decision making (Çakir and Ulukan 2019; Çakir and Ulukan 2020).

In this section we briefly summarize our latest research in text classification. The goal of any text classification process is to assign one or more predetermined categories to classify each of the texts. We are proposing a declarative approach consisting of classifying texts according to a set of predefined categories by using semantic relations and the ability of Bousi~Prolog to weakly unify. The proposed method consists of the following steps:

1. **Knowledge Base Building**: The categories are (semantically) defined by extracting a set of proximity equations from standard thesauri and ontologies (WordNet in our case). The set of proximity equations form the significative subset of the thesaurus or ontology that we will use in the classification process and, by abuse of language, we name it the “ontology” file.

2. **Flexible Search and Computing Occurrence Degrees**: For each document content, the words close to a category are searched in order to classify them, and their degrees of occurrence are obtained. The occurrence degree of a word is an aggregation of the number of occurrences of the word (in a document) and its approximation degree with regard to the category analyzed.

3. **Computing Document Compatibility Degrees**: The compatibility degrees of the documents with regard to a category are computed using a specific compatibility measure. A compatibility measure is an operation which uses the occurrence degrees of the words close to a category to calculate a document compatibility degree, that is, an index of how compatible the document is with regard to the category analysed.

4. **Classification Process**: Finally, each document is classified as pertaining to the category or categories that return a higher compatibility degree. We assign to a document all the categories that have a compatibility index between the maximum compatibility, Max, obtained for that document and a minimum Min=0.9*Max.

It is noteworthy that our approach to text classification does not need a pre-classified set of training documents. The proposed method only requires the category names as user input. Hence, our method is not based on category occurrence frequency, but depends greatly on the definition of that category and how the text fits that definition.

Thanks to the integration with WordNet, we can generate the ontology files without human intervention, starting from the set of predefined categories. Ontology files are computed either by: i) generating several level of hyponyms of a category and obtaining the similarity degree between them and the category by using a similarity measure (PATH, WUP, etc.); or ii) taking the gloss of a category (which can be seen as the definition of the category), extracting a list of words using natural language processing techniques, and then obtaining the degree of relation between them and the category by using a similarity measure.

Once the ontology file is generated and the categories from which we start are defined (in terms of their semantic relationship with other words), we can then apply the remaining steps of our classification algorithm.

An application implementing the method described above can be found at the URL https://dectau.uclm.es/bousi-prolog/applications/, and a preliminary paper on this subject is (Al-Sayadi et al. 2020). The results shown in that paper are encouraging in terms of Precision,
Recall and F-measure. For instance, for the dataset “News Wires-2 (Reuters-10)”, which is a set of short texts (news limited up to 160 characters long) selected from Reuters-21578, we obtain an average Precision, Recall and F-measure of 73.07%, 55.59% and 62.99% respectively. Our immediate goal is to improve Recall and to contribute to provide explainable results.

In order to show the feasibility of integrating WordNet into Bousi-Prolog, we undertook an experimental assessment of the cost of generating several ontologies and classifying some datasets. The results are shown in Tables 4 and 5 with CPU runtime in seconds, the number of inferences performed during the computation, and information on memory consumption in megabytes.

Table 4. Performance of automatic generation of ontologies based on WordNet hyponyms

| Ontology file | Equs. | Runtime (s) | Inferences | Global Stack (Mb) | Local Stack (Mb) |
|---------------|-------|-------------|------------|-------------------|-----------------|
| odp_hyp       | 263   | 0.038       | 287,140    | 0.823461          | 0.512711        |
| enviweb_hyp   | 352   | 0.486       | 399,114    | 0.767675          | 0.723183        |
| reutersshorts_hyp | 278   | 0.039       | 286,338    | 0.348183          | 0.554588        |
| reuters10_hyp | 314   | 0.044       | 328,070    | 0.541951          | 0.618492        |
| AVERAGE       | 302   | 0.152       | 325,165    | 0.620317          | 0.602243        |

Using similarity measures based on information content

| Ontology file | Equs. | Runtime (s) | Inferences | Global Stack (Mb) | Local Stack (Mb) |
|---------------|-------|-------------|------------|-------------------|-----------------|
| odp_hyp_ic    | 263   | 44.020      | 190,808,568| 0.245071          | 0.512710        |
| enviweb_hyp_ic| 352   | 28.529      | 124,793,835| 0.344101          | 0.723183        |
| reutersshorts_hyp_ic | 278   | 1.641       | 7,382,626  | 0.807327          | 0.554588        |
| reuters10_hyp_ic | 314   | 25.871      | 111,821,332| 0.323959          | 0.618492        |
| AVERAGE       | 302   | 25.015      | 108,701,590| 0.430115          | 0.602243        |

Table 5. Performance of classifying datasets from WordNet-generated ontologies

| Dataset                      | Runtime (s) | Inferences | Global Stack (Mb) | Local Stack (Mb) |
|------------------------------|-------------|------------|-------------------|-----------------|
| Web Snippets (ODP): 115      | 1.453       | 14,510,417 | 2.49              | 2.31            |
| documents                    |             |            |                   |                 |
| News Snippets (EnviWeb): 116 | 1.668       | 16,508,815 | 1.72              | 1.71            |
| documents                    |             |            |                   |                 |
| News Wires-1 (Reuters-Short):| 2.766       | 27,308,333 | 3.32              | 4.39            |
| 267 documents                |             |            |                   |                 |
| News Wires-2 (Reuters-10):   | 422.729     | 3,586,006,321| 247.62          | 341.49          |
| 8,599 documents              |             |            |                   |                 |

In Table 4, each row groups the average data obtained when generating ontologies for a given data set using three different similarity measures. The column “Equs.” shows the number of proximity equations generated per ontology. The first part of the table presents data related to similarity measures based on counting edges (PATH, WUP and LCH) while the second part gives those based on information content (RES, JCN, LIN).

Note that, for the ontologies which use similarity measures based on counting edges, the cost

\[10\text{ Precision: percentage of total positive classifications w.r.t. the total of classifications performed by the classifier method. In this case, ‘positive classification’ means a classification where the classifier and the expert judgment co-}
\[11\text{ incide. Recall: percentage of total positive classifications w.r.t. the total of classifications performed by the expert classifier. F-measure: the harmonic mean between precision and recall.}

[http://www.daviddlewis.com/resources/testcollections/reuters21578/]
of generating and storing proximity equations (pairs) in a file 302 is 0.152 seconds, on average. This signifies that for this kind of measure the latency is 0.5 ms/equ and the throughput 1,986.8 equus/s. Similarly, for measures based on information content, the latency is 82.9 ms/equ, and the throughput is 12.1 equus/s.

As can be seen, building ontologies using similarity measures based on information content has a higher cost due to the complexity of this kind of similarity measure (involving the computation of the LCS and the generation of all its hyponyms, to establish its information content, in order to compute the similarity degree of two words).

Table 5 sets out information about the average cost of classifying four different datasets using the previously generated ontologies.

### 7 Conclusions

We have presented techniques to embody the information stored in the lexical database WordNet (Fellbaum 1998; Fellbaum 2006; Miller 1995) into the fuzzy logic programming language Bousi~Prolog (Rubio-Manzano and Julián-Iranzo 2014; Julián-Iranzo and Rubio-Manzano 2017; Julián-Iranzo and Sáenz-Pérez 2018a). However, the techniques developed can be used to connect WordNet to any logic programming language that uses an operational semantics based on some variant of WSLD resolution.

The main contributions of this study are the following:

1. We have implemented, in Prolog, all the usual similarity measures (based on counting edges and on information content) to be found in standard tools such as wordnet:similarity (Pedersen et al. 2004).
2. A whole BPL subsystem (wn-connect) has been developed, providing those measures and several built-in predicates to obtain useful information about words and synsets in WordNet. This subsystem can be used independently in a Prolog interpreter.
3. We have implemented directives to generate proximity equations from a set of words, linking them with an approximation degree. Hence, the significance of this work is to make a fuzzy treatment of concepts via proximity relations possible, and also to endow Bousi~Prolog with linguistic characteristics.
4. Because Bousi~Prolog allows WordNet databases to be accessed easily, it is possible to use interesting relations (antonymy, meronymy, etc.), or to use causal relations, to reason.
5. We have provided the system implementing these techniques as a desktop application (for Windows, Mac and Linux OS’s – dectau.uclm.es/bousi-prolog), and also an online web interface (dectau.uclm.es/bplweb).
6. Finally, we have undertaken an experimental assessment: Firstly, measuring the performance of the implemented WordNet-based similarity measures and the cost of generating hyponymy-based ontologies; and, secondly, executing a text classification application implemented using Bousi~Prolog and its connection to WordNet, concluding that Bousi~Prolog has a reasonable performance w.r.t. other systems.

As future work, experiments suggest enhancing the performance of Bousi~Prolog by introducing memorizing techniques, and optimizing their compilation by leveraging tail recursion optimization, and also, implementing relatedness measures based on other techniques such as word embeddings.
Acknowledgements

We would like to express our gratitude to the anonymous reviewers and the area editor for their valuable comments that have greatly improved the final version of our paper.

References

AL-SAYADI, S. H., JULIÁN-IRANZO, P., ROMERO, F. P., AND SÁENZ-PÉREZ, F. 2020. A Fuzzy Declarative Approach to Classify Unlabeled Short Texts based on Automatically Constructed WordNet Ontologies. In Proc. of the 12th European Symposium on Computational Intelligence and Mathematics, ESCIM 2020, 1–6.

ALLAHYARI, M., POURIYEH, S., ASSEFI, M., SAFAEI, S., TRIPPE, E., GUTIERREZ, J., AND KOCHUT, K. 2017. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. CoRR abs/1707.02919.

BAEZA-YATES, R. AND RIBEIRO-NETO, B. 2011. Modern Information Retrieval - The Concepts and Technology behind Search, Second Edition. Pearson Education Ltd., Harlow, England.

BUDANITSKY, A. AND HIRST, G. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics 32, 1, 13–47.

ÇAKIR, E. AND ULUKAN, H. Z. 2019. A Fuzzy Logic Programming Environment for Recycling Facility Selection. In Proc. of the 11th International Joint Conference on Computational Intelligence, IJCCI 2019. ScitePress, 367–374.

ÇAKIR, E. AND ULUKAN, H. Z. 2020. A Fuzzy Linguistic Programming for Sustainable Ecotourism Activities. In Proc. of the 10th Annual Computing and Communication Workshop and Conference, CCWC 2020. IEEE, 121–126.

FELDBAUM, C. 1998. WordNet: An Electronic Lexical Database. MIT Press.

FELDBAUM, C. 2006. WordNet(s). In Encyclopedia of Language & Linguistics, Second Edition, K. B. E. in Chief), Ed. Vol. 13. Elsevier, Oxford., 665–670.

FELDBAUM, C. AND et al. 2006. WordNet File Formats: prologdb(5WN). https://wordnet.princeton.edu/documentation/prologdb5wn.

FONTANA, F. A. AND FORMATO, F. 1999. Likelog: A Logic Programming Language for Flexible Data Retrieval. In Proc. of the 1999 ACM Symposium on Applied Computing (SAC’99). 260–267.

FONTANA, F. A. AND FORMATO, F. 2002. A Similarity-based Resolution Rule. International Journal of Intelligent Systems 17, 9, 853–872.

FRANCIS, W. N. AND KUCERA, H. 1964, 1971, 1979. A Standard Corpus of Present-Day Edited American English, for use with Digital Computers (Brown). https://www.sketchengine.eu/brown-corpus/

JIANG, J. J. AND CONRATH, D. W. 1997. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In Proc. of the 10th Research on Computational Linguistics International Conference, ROCLING 1997. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP), 19–33.

JULIÁN-IRANZO, P. AND RUBIO-MANZANO, C. 2015. Proximity-based Unification Theory. Fuzzy Sets and Systems 262, 21–43.

JULIÁN-IRANZO, P. AND RUBIO-MANZANO, C. 2017. A Sound and Complete Semantics for a Similarity-based Logic Programming Language. Fuzzy Sets and Systems 317, 1–26.

JULIÁN-IRANZO, P. AND SÁENZ-PÉREZ, F. 2017. FuzzyDES and how DES met Bousi Prolog. In Proc. of the 2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2017, 1–6.

JULIÁN-IRANZO, P. AND SÁENZ-PÉREZ, F. 2018a. An Efficient Proximity-based Unification Algorithm. In Proc. of the 2018 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2018. 1–8.

JULIÁN-IRANZO, P. AND SÁENZ-PÉREZ, F. 2018b. A Fuzzy Datalog Deductive Database System. IEEE Transactions on Fuzzy Systems 26, 2634–2648.

JULIÁN-IRANZO, P. AND SÁENZ-PÉREZ, F. 2019. WordNet and Prolog: why not? In Proc. of the 11th Conference of the European Society for Fuzzy Logic and Technology, EUSFLAT 2019. 1–8.
Implementing WordNet Lexical Similarity Measures

LEACOCK, C. AND CHODOROW, M. 1998. Combining Local Context and Wordnet Similarity for Word Sense Identification. In WordNet: An Electronic Lexical Database, C. Fellbaum, Ed. MIT Press., 265–283.

LEE, R. 1972. Fuzzy Logic and the Resolution Principle. Journal of the ACM 19, 1, 119–129.

LIN, D. 1998. An Information-Theoretic Definition of Similarity. In Proc. of the 15th International Conference on Machine Learning. Morgan Kaufmann, 296–304.

LOIA, V., SENATORE, S., AND SESSA, M. I. 2001. Similarity-based SLD Resolution and Its Implementation in an Extended Prolog System. In Proc. of the 2001 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2001. 650–653.

MARTELLI, A. AND MONTANARI, U. 1982. An Efficient Unification Algorithm. ACM Transactions on Programming Languages and Systems 4, 258–282.

MILLER, G. A. 1995. WordNet: A Lexical Database for English. Comms. of the ACM 38, 11, 39–41.

MILLER, G. A., LEACOCK, C., TENG, R., AND BUNKER, R. 1993. A Semantic Concordance. In Proc. of the Workshop on Human Language Technology, HLT 1993. 303–308.

PEDERSEN, T., PATWARDHAN, S., AND MICHELIZZI, J. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In Proc. of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence. AAAI Press / The MIT Press, 1024–1025.

RESNIK, P. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, 2 Volumes. Morgan Kaufmann, 448–453.

ROMERO, F. P., JUÁN-IRANZO, P., FERREIRA-SATLER, M., AND GALLARDO-CASERO, J. 2013. Classifying Unlabeled Short Texts using a Fuzzy Declarative Approach. Language Resources and Evaluation 47, 1, 151–178.

RUBIO-MANZANO, C. AND JUÁN-IRANZO, P. 2014. Fuzzy Linguistic Prolog and its Applications. Journal of Intelligent and Fuzzy Systems 26, 1503–1516.

RUBIO-MANZANO, C. AND JUÁN-IRANZO, P. 2015. Incorporation of Abstraction Capability in a Logic-based Framework by using Proximity Relations. Journal of Intelligent and Fuzzy Systems 29, 4, 1671–1683.

RUBIO-MANZANO, C. AND TRIVIÑO, G. 2016. Improving Player Experience in Computer Games by using Players’ Behavior Analysis and Linguistic Descriptions. International Journal of Human-Computer Studies 95, 27 – 38.

SANTUS, E., WANG, H., CHERSONI, E., AND ZHANG, Y. 2018. A Rank-Based Similarity Metric for Word Embeddings. In Proc. of the 56th Annual Meeting of the Association for Computational Linguistics ACL 2018 (2). Association for Computational Linguistics, 552–557.

SERRANO-GUERRERO, J., OLIVAS, J. A., ROMERO, F. P., AND HERRERA-VIEDMA, E. 2015. Sentiment Analysis: A Review and Comparative Analysis of Web Services. Inf. Sci. 311, 18–38.

SESSA, M. I. 2002. Approximate Reasoning by Similarity-based SLD Resolution. Theoretical Computer Science 275, 1-2, 389–426.

VAN EMDE, M. AND KOWALSKI, R. 1976. The Semantics of Predicate Logic as a Programming Language. Journal of the ACM 23, 4, 733–742.

WU, Z. AND PALMER, M. S. 1994. Verb Semantics and Lexical Selection. In Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics. Morgan Kaufmann Publishers / ACL, 133–138.

ZADEH, L. 1965. Fuzzy Sets. Information and Control 8, 338–353.