USP-IBM-1 and USP-IBM-2: The ILP-based Systems for Lexical Sample WSD in SemEval-2007

Lucia Specia, Maria das Graças Volpe Nunes
ICMC - University of São Paulo
Trabalhador São-Carlense, 400, São Carlos, 13560-970, Brazil
{lspecia, gracan}@icmc.usp.br

Ashwin Srinivasan, Ganesh Ramakrishnan
IBM India Research Laboratory
Block 1, Indian Institute of Technology, New Delhi 110016, India
{ashwin.srinivasan, ganramkr}@in.ibm.com

Abstract

We describe two systems participating of the English Lexical Sample task in SemEval-2007. The systems make use of Inductive Logic Programming for supervised learning in two different ways: (a) to build Word Sense Disambiguation (WSD) models from a rich set of background knowledge sources; and (b) to build interesting features from the same knowledge sources, which are then used by a standard model-builder for WSD, namely, Support Vector Machines. Both systems achieved comparable accuracy (0.851 and 0.857), which outperforms considerably the most frequent sense baseline (0.787).

1 Introduction

Word Sense Disambiguation (WSD) aims to identify the correct sense of ambiguous words in context. Results from the last edition of the Senseval competition (Mihalcea et al., 2004) have shown that, for supervised learning, the best accuracies are obtained with a combination of various types of features, together with traditional machine learning algorithms based on feature-value vectors, such as Support Vector Machines (SVMs) and Naive Bayes. While the features employed by these approaches are mostly considered to be "shallow", that is, extracted from corpus or provided by shallow syntactic tools like part-of-speech taggers, it is generally thought that significant progress in automatic WSD would require a "deep" approach in which access to substantial body of linguistic and world knowledge could assist in resolving ambiguities. Although the access to large amounts of knowledge is now possible due to the availability of lexicons like WordNet, parsers, etc., the incorporation of such knowledge has been hampered by the limitations of the modelling techniques usually employed for WSD. Using certain sources of information, mainly relational information, is beyond the capabilities of such techniques, which are based on feature-value vectors. Arguably, Inductive Logic Programming (ILP) systems provide an appropriate framework for dealing with such data: they make explicit provisions for the inclusion of background knowledge of any form; the richer representation language used, based on first-order logic, is powerful enough to capture contextual relationships; and the modelling is not restricted to being of a particular form (e.g., classification).

We describe the investigation of the use of ILP for WSD in the Lexical Sample task of SemEval-2007 in two different ways: (a) the construction of models that can be used directly to disambiguate words; and (b) the construction of interesting features to be used by a standard feature-based algorithm, namely, SVMs, to build disambiguation models. We call the systems resulting of the two different approaches "USP-IBM-1" and "USP-IBM-2", respectively. The background knowledge is from 10 different sources of information extracted from corpus, lexical resources and NLP tools.

In the rest of this paper we first present the specification of ILP implementations that construct ILP models and features (Section 2) and then describe the experimental evaluation on the SemEval-2007 Lexical Sample task data (Section 3).

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 442–445,
Prague, June 2007.

2 Inductive Logic Programming

Inductive Logic Programming (ILP) (Muggleton, 1991) employs techniques from Machine Learning and Logic Programming to build first-order theories or descriptions from examples and background knowledge, which are also represented by first-order clauses. Functionally, ILP can be characterised by two classes of programs. The first, predictive ILP, is concerned with constructing models (in this case, sets of rules) for discriminating accurately amongst positive and negative examples. The partial specifications provided by (Muggleton, 1994) form the basis for deriving programs in this class:

- B is background knowledge consisting of a finite set of clauses $= \{C_1, C_2, \ldots\}$
- E is a finite set of examples $= E^+ \cup E^-$ where:
  - *Positive Examples.* $E^+ = \{e_1, e_2, \ldots\}$ is a non-empty set of definite clauses
  - *Negative Examples.* $E^- = \{\overline{f_1}, \overline{f_2} \ldots\}$ is a set of Horn clauses (this may be empty)
- H, the output of the algorithm given B and E, is acceptable if these conditions are met:
  - *Prior Satisfiability.* $B \cup E^- \not\models \square$
  - *Posterior Satisfiability.* $B \cup H \cup E^- \not\models \square$
  - *Prior Necessity.* $B \not\models E^+$
  - *Posterior Sufficiency.* $B \cup H \models e_1 \wedge e_2 \wedge \ldots$

The second category of ILP programs, descriptive ILP, is concerned with identifying relationships that hold amongst the background knowledge and examples, without a view of discrimination. The partial specifications for programs in this class are based on the description in (Muggleton and Raedt, 1994):

- B is background knowledge
- E is a finite set of examples (this may be empty)
- H, the output of the algorithm given B and E is acceptable if the following condition is met:
  - *Posterior Sufficiency.* $B \cup H \cup E \not\models \square$

The intuition behind the idea of exploiting a feature-based model constructor that uses first-order features is that certain sources of structured information that cannot be represented by feature vectors can, by a process of "propositionalization", be identified and converted in a way that they can be accommodated in such vectors, allowing for traditional learning techniques to be employed. Essentially, this involve two steps: (1) a feature-construction step that identifies all the features, that is, a set of clauses H, that are consistent with the constraints provided by the background knowledge B (descriptive ILP); and (2) a feature-selection step that retains some of the features based on their utility in classifying the examples, for example, each clause must entail at least one positive example (predictive ILP). In order to be used by SVMs, each clause h_i in H is converted into a boolean feature f_i that takes the value 1 (or 0) for any individual for which the body of the clause is true (if the body is false). Thus, the set of clauses H gives rise to a boolean vector for each individual in the set of examples. The features constructed may express conjunctions on different knowledge sources. For example, the following boolean feature built from a clause for the verb "ask" tests whether the sentence contains the expression "ask out" and the word "dinner". More details on the specifications of predictive and descriptive ILP for WSD can be found in (Specia et al., 2007):

$$f_1(X) = \begin{cases} 1 & expr(X, \text{'ask out'}) \wedge bag(X, dinner) = 1 \\ 0 & \text{otherwise} \end{cases}$$

3 Experiments

We investigate the performance of two kinds of ILP-based models for WSD:

1. *ILP models* (USP-IBM-1 system): models constructed by an ILP system for predicting the correct sense of a word.
2. *ILP-assisted models* (USP-IBM-2 system): models constructed by SVMs for predicting the correct sense of a word that, in addition to existing shallow features, use features built by an ILP system according to the specification for feature construction in Section 2.

The data for the English Lexical Sample task in SemEval-2007 consists of 65 verbs and 35 nouns. Examples containing those words were extracted from the WSJ Penn Treebank II and Brown corpus. The number of training / test examples varies from 19 / 2 to 2,536 / 541 (average = 222.8 / 48.5). The senses of the examples were annotated according to OntoNotes tags, which are groupings of WordNet senses, and therefore are more coarse-grained. The number of senses used in the training examples for a given word varies from 1 to 13 (average = 3.6).

First-order clauses representing the following background knowledge sources, which were automatically extracted from corpus and lexical resources or provided by NLP tools, were used to describe the target words in both systems:

B1. Unigrams consisting of the 5 words to the right and left of the target word.

B2. 5 content words to the right and left of the target word.

B3. Part-of-speech tags of 5 words to the right and left of the target word.

B4. Syntactic relations with respect to the target word. If that word is a verb, subject and object syntactic relations are represented. If it is a noun, the representation includes the verb of which it is a subject or object, and the verb / noun it modifies.

B5. 12 collocations with respect to the target word: the target word itself, 1st preposition to the right, 1st and 2nd words to the left and right, 1st noun, 1st adjective, and 1st verb to the left and right.

B6. A relative count of the overlapping words in the sense inventory definitions of each of the possible senses of the target word and the words surrounding that target word in the sentence, according to the sense inventories provided.

B7. If the target word is a verb, its selectional restrictions, defined in terms of the semantic features of its arguments in the sentence, as given by LDOCE. WordNet relations are used to make the verification more generic and a hierarchy of feature types is used to account for different levels of specificity in the restrictions.

B8. If the target word is a verb, the phrasal verbs possibly occurring in a sentence, according to the list of phrasal verbs given by dictionaries.

B9. Pairs of words in the sentence that occur frequently in the corpus related by verb-subject/object or subject/verb/object-modifier relations.

B10. Bigrams consisting of adjacent words in a sentence occurring frequently in the corpus.

Of these 10 sources, B1–B6 correspond to the so called "shallow features", in the sense that they can be straightforwardly represented by feature vectors. A feature vector representation of these sources is built to be used by the feature-based model constructor. Clausal definitions for B1–B10 are directly used by the ILP system.

We use the Aleph ILP system (Srinivasan, 1999) to construct disambiguation models in USP-IBM-1 and to construct features to be used in USP-IBM-2. Feature-based model construction in USP-IBM-2 system is performed by a linear SVM (the SMO implementation in WEKA).

In the USP-IBM-1 system, for each target word, equipped with examples and background knowledge definitions (B1–B10), Aleph constructs a set of clauses in line with the specifications for predictive ILP described in Section 2. Positive examples are provided by the correct sense of the target word. Negative examples are generated automatically using all the other senses. 3-fold cross-validation on the training data was used to obtain unbiased estimates of the predictive accuracy of the models for a set of relevant parameters. The best average accuracies were obtained with the greedy induction strategy, in conjunction with a minimal clause accuracy of 2. The constructed clauses were used to predict the senses in the test data following the order of their production, in a decision-list like manner, with the addition to the end of a default rule assigning the majority sense for those cases which are not covered by any other rule.

In the USP-IBM-2 system, for constructing the "good" features for each target word from B1–B10 (the "ILP-based features"), we first selected, in Aleph, the clauses covering at least 1 positive example. 3-fold cross-validation on the training data was performed in order to obtain the best model possible using SVM with features in B1–B6 and the ILP-based features. A feature selection method based on information gain with various percentages of features to be selected ($1/64$, ..., $1/2$) was used, which resulted in different numbers of features for each target word.

	Baseline	USP-IBM-1	USP-IBM-2
Nouns	0.809	0.882	0.882
Verbs	0.762	0.817	0.828
All	0.787	0.851	0.857

Table 1: Average accuracies of the ILP-based models for different part-of-speeches

sense(X, 3) :-
expr(X, 'come to').

sense(X, 1) :-
satisfy_restrictions(X, [animate], nil);
(relation(X, subj, B), pos(X, B, nnp)).

Figure 1: Examples of rules learned for "come"

Table 1 shows the average accuracy of a baseline classifier that simply votes for the most frequent sense of each word in the training data against the accuracy of our ILP-based systems, USP-IBM-1 and USP-IBM-2, according to the part-of-speech of the target word, and for all words. Clearly, the "majority class" classifier performs poorest, on average. The difference between both ILP-based systems and the baseline is statistically significant according to a paired t-test with $p < 0.01$. The two ILP-based models appear to be comparable in their average accuracy. Discarding ties, IBM-USP-2 outperforms IBM-USP-1 for 31 of the words, but the advantage is not statistically significant (cf. paired t-test).

The low accuracy of the ILP-based systems for certain words may be consequence of some characteristics of the data. In particular, the sense distributions are very skewed in many cases, with different distributions in the training and test data. For example, in the case of "care" (accuracy = 0.428), the majority sense in the training data is 1 (78.3%), while in the test data the majority sense is 2 (71%). In cases like this, many of the test examples remain uncovered by the rules produced by the ILP system and backing off to the majority sense also results in a mistake, since the majority sense in the training data does not apply for most of the test examples. The same goes for the feature-based system: features which are relevant for the test examples will not be built or selected.

One relevant feature of ILP is its ability to produce expressive symbolic models. These models can reproduce any kind of background knowledge using sets of rules testing conjunctions of different types of knowledge, which may include variables (intensional clauses). This is valid both for the construction of predictive models and for the construction of features (which are derived from the clauses). Examples of rules induced for the verb "come" are given in Figure 1. The first rule states that the sense of the verb in a sentence X will be 3 (progress to a state) if that sentence contains the expression "come to". The second rule states that the sense of the verb will be 1 (move, travel, arrive) if its subject is "animate" and there is no object, or if it has has a subject B that is a proper noun (nnp).

4 Concluding Remarks

We have investigated the use of ILP as a mechanism for incorporating shallow and deep knowledge sources into the construction of WSD models for the Semeval-2007 Lexical Sample Task data. Results consistently outperform the most frequent sense baseline. It is worth noticing that the knowledge sources used here were initially designed for the disambiguation of verbs (Specia et al., 2007) and therefore we believe that further improvements could be achieved with the identification and specification of other sources which are more appropriate for the disambiguation of nouns.

References

R. Mihalcea, T. Chklovski, A. Kilgariff. 2004. The SENSEVAL-3 English Lexical Sample Task. *SENSEVAL-3: 3rd Int. Workshop on the Evaluation of Systems for Semantic Analysis of Text*, 25–28.

S. Muggleton. 1991. Inductive Logic Program-ming. *New Generation Computing*, 8(4):29-5-318.

S. Muggleton. 1994. Inductive Logic Programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11.

S. Muggleton and L. D. Raedt. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679.

L. Specia, M.G.V. Nunes, A. Srinivasan, G. Ramakrishnan. 2007. Word Sense Disambiguation using Inductive Logic Programming. *Proceedings of the 16th International Conference on ILP*, Springer-Verlag.

A. Srinivasan. 1999. *The Aleph Manual*. Computing Laboratory, Oxford University.