An Investigation into Feature Construction to Assist Word Sense Disambiguation

Lucia Specia[1], Ashwin Srinivasan[2,3], Ganesh Ramakrishnan[2], Sachindra Joshi[2], and Maria das Graças Volpe Nunes[4]

[1] Xerox Research Centre Europe,
6 Chemin de Maupertuis, Meylan 38240, France.
lucia.specia@xrce.xerox.com
[2] IBM India Research Laboratory,
4-C, Institutional Area,
Vasant Kunj, New Delhi 110 070.
ashwin.srinivasan,jsachind,ganramkr@in.ibm.com
[3] Department of CSE & Centre for Health Informatics,
University of New South Wales, Sydney.
[4] ICMC - Universidade de São Paulo, Trabalhador São-Carlense, 400, São Carlos, 13560-970, Brazil.
gracan@icmc.usp.br

Abstract. Identifying the correct sense of a word in context is crucial for many tasks in natural language processing (machine translation is an example). State-of-the art methods for Word Sense Disambiguation (WSD) build models using hand-crafted features that usually capturing shallow linguistic information. Complex background knowledge, such as semantic relationships, are typically either not used, or used in specialised manner, due to the limitations of the feature-based modelling techniques used. On the other hand, empirical results from the use of Inductive Logic Programming (ILP) systems have repeatedly shown that they can use diverse sources of background knowledge when constructing models. In this paper, we investigate whether this ability of ILP systems could be used to improve the predictive accuracy of models for WSD. Specifically, we examine the use of a general-purpose ILP system as a method to construct a set of features using semantic, syntactic and lexical information. This feature-set is then used by a common modelling technique in the field (a support vector machine) to construct a classifier for predicting the sense of a word. In our investigation we examine one-shot and incremental approaches to feature-set construction applied to monolingual and bilingual WSD tasks. The monolingual tasks use 32 verbs and 85 verbs and nouns (in English) from the SENSEVAL-3 and SemEval-2007 benchmarks; while the bilingual WSD task consists of 7 highly ambiguous verbs in translating from English to Portuguese. The results are encouraging: the ILP-assisted models show substantial improvements over those that simply use shallow features. In addition, incremental feature-set construction appears to identify smaller and better sets of features. Taken together, the results suggest that the use of ILP with diverse sources of background knowledge provide a way for making substantial progress in the field of WSD.

1 Introduction

Word Sense Disambiguation (WSD) aims to identify the correct sense of an ambiguous word in a sentence. Usually described as an "intermediate task" [42], it is necessary in many natural language tasks like machine translation, information retrieval, question answering, and so on. That it is extremely difficult to completely solve WSD is a long-standing view [2] and accuracies with state-of-the art methods are substantially lower than in other areas of text understanding. Part-of-speech tagging accuracies, for example, are now over 95%; in contrast, the best WSD results for fine-grained sense distinctions are still below 80%.

The principal approach adopted for the automatic construction of WSD models is a "shallow" one. In this, sample data consisting of sentences with the ambiguous words and their correct sense are represented using features capturing some limited context around the ambiguous words in each sentence. For example, features may denote words on either side of an ambiguous word and the part-of-speech tags of those words. Sample data represented in this manner are then used by a statistical model constructor to build a general predictive model for disambiguating words. Results from the literature on benchmark data like those provided under the various SENSEVAL competitions[5] suggest that support vector machines (SVMs) yield models with one of the highest accuracies. As this competition shows, some improvements have been achieved in the accuracy of predictions, despite of the use of limited, shallow information sources. On the other hand, 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. However, the incorporation of large amounts of domain knowledge has been hampered by the following: (a) access to such information in electronic form suitable for constructing models; and (b) modeling techniques capable of utilizing diverse sources of domain knowledge. The first of these difficulties is now greatly alleviated by the availability in electronic form of very large semantic lexicons like WordNet [19], dictionaries, parsers, grammars and so on. In addition, there are now very large amounts of "shallow" data in the form of electronic text corpora from which statistical information can be readily extracted. Using these diverse sources of information is, however, beyond the capabilities of existing general-purpose statistical methods that have been used for WSD. Arguably, Inductive Logic Programming (ILP) systems provide a more general-purpose framework for dealing with such data: there are explicit provisions made for the inclusion of background knowledge of any form; the representation language is powerful enough to capture the contextual relationships that arise; and modeling is not restricted to being of a particular form (for example, classification only).

In this paper, we investigate the possibility that an ILP system equipped with deep and shallow knowledge sources can be used to improve the predictivity of models for WSD. Specifically, our interest is in ILP as

[5] see: http://www.senseval.org

a technique for constructing new features, and our hypothesis is the following:

> **ILP-features hypothesis.** Feature-construction by ILP provides a general method of introducing background knowledge for improving the predictive accuracy of WSD models.

Empirical evidence is sought by investigating the use of ILP-based feature-construction on 124 datasets from three substantial WSD tasks (thus testing general applicability). When constructing features, an ILP system is provided with background knowledge drawn from 10 different sources (thus testing the ability to use a diverse range of lexical, syntactic and semantic information). In all cases, we find that when hand-crafted features are augmented by ILP-constructed features (we call these "ILP-assisted models"), the predictive power of models *does* improve significantly.
The rest of the paper is organized as follows. In Section 2 we present some related work on WSD. A specification for ILP implementations that construct features for use in ILP-assisted models is in Section 3.1. We then describe an implementation that meets these specifications in Section 3.2. The empirical evaluation comprising our investigation is described in Section 4. This includes materials (Section 4.1) and methods (Section 4.2). A summary of results and a related discussion are in Section 5. Section 6 concludes the paper. The paper is accompanied by an appendix that contains detailed tabulations of the experimental results.

2 Models for Word Sense Disambiguation

The earliest computer-executable models for WSD are manually constructed, capturing specific aspects of human disambiguation expertise in symbolic structures like semantic networks [30] and semantic frames [7, 17]. Early reports also exist of sub-symbolic neural networks [5]. Most of these techniques appear to have suffered from the important difficulty in manual acquisition of expert knowledge, resulting in their application being limited to very small subsets of the languages.
The development of machine readable resources like lexical databases, dictionaries and thesauri has provided a turning point in WSD, enabling the development of techniques that used linguistic and extra-linguistic information extracted automatically from these resources [1, 15, 41]. While the resources provided ready access to large bodies of knowledge, the actual disambiguation models continued to be manually codified. This changed with the use of statistical and machine-learning techniques for constructing models. The characteristic of these methods is the use of a corpus of examples of disambiguation to construct automatically models for disambiguation [27, 32, 43]. The most common of these "corpus-based" techniques employ statistical methods that construct models based on features representing frequencies estimated from a corpus. For example, these may be the frequencies of some words on either side of the ambiguous word. While techniques using such "shallow" features that refer to the local context of the ambiguous word have

yielded better models than the previous ones, the accuracies obtained are still low, and significant improvements do not appear to be forthcoming. More sophisticated corpus-based approaches such as [40] try to incorporate deeper knowledge using machine readable resources. These are special-purpose methods aimed at specific tasks and it is not clear how they could be scaled-up for use across a wide range of WSD tasks. ILP provides a general-purpose approach that can be tailored to a variety of NLP tasks by the incorporation of appropriate background knowledge. The use of ILP to build WSD models was first investigated by Specia [34, 35]. Preliminary work on the use of ILP to construct features to be used by standard propositional machine learning algorithms like SVMs can be found in [36, 37]. The work here extends these substantially by exploring alternate ways of using ILP to build features and in terms of experimental results.

3 Feature Construction with ILP

We motivate the task of feature-construction using the "trains" problem, originally proposed by Ryzhard Michalski. The task is to construct a model that can discriminate between eastbound and westbound trains, using properties of their carriages, and the loads carried (see Fig. 1).

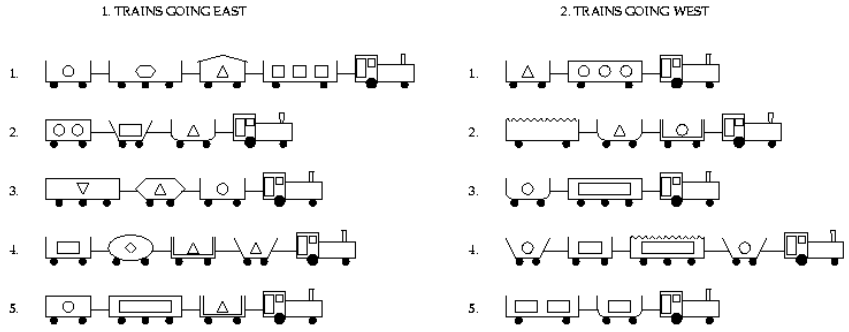

Fig. 1. The trains problem. Trains are classified either as "eastbound" or "westbound". They have open or closed carriages of different shapes, lengths, and so on. The carriages contain loads of different shapes and numbers. The task is to construct a model that, given the description of a train, can predict whether it will be eastbound or westbound.

We will assume that the trains can be adequately described by background predicates that will become evident shortly. Further, let us assume that the 10 trains shown in the figure are denoted $t1, t2, \ldots, t10$ and that their classifications are encoded as a set of logical statements of the form: $Class(t1, eastbound), Class(t2, eastbound), \ldots, Class(t10, westbound)$.

As is quite normal in the use of ILP for feature-construction, we will assume features to be Boolean valued, and obtained from some clause identified by the ILP program. For example, Fig. 2 shows five such features, found by an ILP engine, and the corresponding tabular representation of the 10 examples in Fig. 1.

$f1 = \begin{cases} 1 \; iff \; has\ short\ and\ closed\ car \\ 0 \; otherwise \end{cases}$

$f2 = \begin{cases} 1 \; iff \; has\ closed\ car\ and\ contains\ triangle\ load \\ 0 \; otherwise \end{cases}$

$f3 = \begin{cases} 1 \; iff \; has\ double-wall\ car\ and\ contains\ triangle\ load \\ 0 \; otherwise \end{cases}$

$f4 = \begin{cases} 1 \; iff \; has\ car\ with\ jagged\ roof \\ 0 \; otherwise \end{cases}$

$f5 = \begin{cases} 1 \; iff \; has\ long\ car\ and\ contains\ rectaglular\ load \\ 0 \; otherwise \end{cases}$

$f6 = \begin{cases} 1 \; iff \; has\ long\ car\ and\ can\ with\ jagged\ roof \\ 0 \; otherwise \end{cases}$

f1	f2	f3	f4	f5	f6	Class
1	1	0	0	0	0	East
1	0	0	0	0	0	East
1	1	0	0	0	0	East
1	0	1	0	0	0	East
1	0	1	0	0	0	East
0	0	0	0	0	0	West
0	0	0	1	0	1	West
0	0	0	0	1	0	West
0	0	0	1	1	1	West
0	0	0	0	0	0	West

Fig. 2. Some Boolean features for the trains problem, and a corresponding tabular representation of the trains in Fig. 1.

Suppose that these 6 features are the only features that can be constructed by the ILP engine, and further, that it is our task to find the best subset of these that can result in the best model. Clearly, if we simply evaluated models obtained with each of the 63 subsets of the set $\{f1, f2, f3, f4, f5, f6\}$ and return the subset that returned the best model, we would be done. Now, let us consider a more practical situation. Suppose the features that can be constructed by an ILP system are not in the 10s, but in the 1000s or even 100s of 1000s. This would make it intractable to construct models with all possible subsets of features. Further, suppose that constructing each feature is not straightforward, computationally speaking, making it impractical to even use the ILP engine to construct all the possible features in the first place. Are we able to nevertheless determine the subset that would yield the best model (which we will now interpret to mean the model with the highest classification accuracy)? The conceptual problem to be addressed is shown in Fig. 3.

Readers will recognise this as somewhat similar to the problem addressed by a randomised procedure for distribution-estimation like Gibb's sampling. There, if the $\mathcal{F}$ features are given (or at least can be enumerated), then the sampling procedure attempts to converge on the best-performing subset without examining the entire space of $2^{|\mathcal{F}|}$ elements. Clearly, if we are unable to generate all possible features in $\mathcal{F}$ beforehand, we are not in a position to use these methods. We first propose a minimal specification to be satisfied by any ILP-based feature constructor.

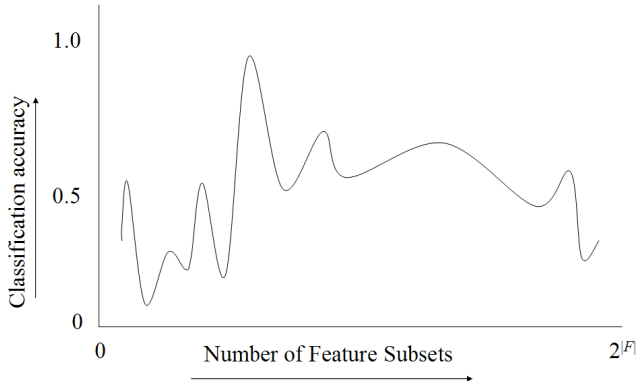

Fig. 3. Identifying the best subset of features for a model-construction algorithm A. The X-axis enumerates the different subsets of features that can be constructed by an ILP engine ($\mathcal{F}$ denotes the set of all possible features that can be constructed by the engine). The Y-axis shows the probability that an an instance drawn randomly using some pre-specified distribution will be correctly classified by a model constructed by A, given the corresponding subset on the X-axis. We wish to identify the subset $k*$ that yields the highest probability, without actually constructing all the features in $\mathcal{F}$.

3.1 Specification

Functionally, ILP can bee largely characterised by two classes of programs. The first, predictive ILP, is concerned with constructing models (sets of rules; or first-order variants of classification or regression trees) for discriminating accurately amongst two sets of examples ("positive" and "negative"). The partial specifications provided by [21] have formed the basis for deriving programs in this class, and are shown in Fig. 4 (we refer the reader to [24] for definitions of the logical terms used).

- 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*. $E^+ = \{p_1, p_2, \ldots\}$ is a non-empty set of definite clauses;
  - *Negative Examples*. $E^- = \{\overline{n_1}, \overline{n_2} \ldots\}$ is a set of Horn clauses (this may be empty)
- H, the output of the algorithm given B and E is acceptable if the following 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*. $B \cup H \models e_1 \wedge e_2 \wedge \ldots$

Fig. 4. A partial specification for a predictive ILP system from [21].

The second category of ILP systems, 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 [22], and are shown in Fig. 5.

- B is background knowledge consisting of a finite set of clauses $= \{C_1, C_2, \ldots\}$
- 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. $B \cup H \cup E \not\models \square$

Fig. 5. A partial specification for a descriptive ILP system based on [22].

The idea of exploiting a feature-based model constructor that uses first-order features can be traced back at least to the LINUS program [14]. More recently, the task of identifying good features using a first-order logic representation has been the province of programs developed under the umbrella of "propositionalization" (see [11] for a review). Programs in this class are not easily characterised as either predictive or descriptive ILP. Conceptually, solutions involve two steps: (1) a feature-construction step that identifies (within computational reason) all the features that are consistent with the constraints provided by the background knowledge. This is characteristic of a descriptive ILP program; and (2) a feature-selection step that retains some of the features based on their utility in classifying the examples. This is characteristic of a predictive ILP program. To this extent, we would like specifications for feature construction to reflect a combination of these two dominant flavours of ILP programs. In particular, we will assume the following:

1. Examples are taken to be some subset of the binary relation $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the set of individuals and $\mathcal{Y}$ some finite set of classes. We therefore do not make any special distinction between positive and negative examples.
2. Given examples of the form described, an ILP system identifies one or more definite clauses of the form $h_i : Class(x, c) \leftarrow Cp_i(x)$, where x is a variable and c is some class in the set of classes $\mathcal{Y}$. Here, adopting terminology from [31], $Cp_i : \mathcal{X} \mapsto \{0, 1\}$ is a "context predicate" that corresponds to a conjunction of literals that evaluates to $TRUE$ (1) or $FALSE$ (0) for any particular individual. We will require that Cp_i contains at least one literal: in logical terms, we therefore require the h_i to be definite clauses with at least two literals (let us call them "mixed definite clauses", to denote the requirement for exactly one positive literal and at least one negative literal). To ensure that the clauses found are not trivial, we require each h_i to entail at least one example in E.
3. Features are conjunctions of literals. Given a clause $h_i : Class(x, c) \leftarrow Cp_i(x)$, found by the ILP system, we construct a one-to-one mapping to a feature f_i as follows: $f_i(x) = 1$ iff $Cp_i(x) = 1$ (and 0

otherwise).[6] An example from a WSD task studied in this paper is shown in Fig. 7.

4. We would like features identified to be relevant. The notion of relevance of a feature in a set $F = \{f_1, \ldots, f_k\}$ can be captured by a probabilistic statement on feature-based representation of the examples. Let individuals be elements of the set $F_1 \times \cdots \times F_k$ (where F_i is the set of values of the i^{th} feature: for us all the F_i are $\{0, 1\}$); and examples be elements of $F_1 \times \cdots \times F_k \times \mathcal{Y}$. Then, as in [9], (with some abuse of notation) we can distinguish between the *strong relevance* of a feature f_i in F, if $Pr(Y|F) \neq Pr(Y|F - \{f_i\})$; and its *weak relevance* if there is some $F' \subset F$ such that $f_i \in F'$ and f_i is strongly relevant in F'. That is, $Pr(Y|F') \neq Pr(Y|F' - \{f_i\})$. As a minimal requirement, we would like an ILP-based feature-constructor to identify features that are at least weakly relevant.

With these assumptions, a partial specification for an ILP-based feature constructor is shown in Fig. 6.

- B is background knowledge consisting of a finite subset of Horn clauses $= \{C_1, C_2, \ldots\}$ drawn from some set $\mathcal{B}$.
- E is a finite set of examples of the form $Class(i, c)$ where i is a member of a set of individuals $\mathcal{X}$ and c is a member of a set of classes $\mathcal{Y}$.
- $\mathcal{H}$ is the set of mixed definite clauses of the form $h_i : Class(x, c) \leftarrow Cp_i(x)$, constructible with predicates, functions and constants in $B \cup E$; $\mathcal{F}$ the set of conjunctions of literals that can be constructed with predicates, functions and constants in $B \cup E$.
- $F = \{f_1, f_2, \ldots, f_k\} \subseteq \mathcal{F}$, the output of the algorithm given B and E is a finite set of features constructed from a set of clauses $H = \{h_1, h_2, \ldots\} \subseteq \mathcal{H}$ if the following condition is met:

  Posterior Sufficiency.
  - For each $h_i \in H$, $B \cup \{h_i\} \models e_1 \vee e_2 \vee \ldots$, where $\{e_1, e_2, \ldots\} \subseteq E$
  - $f_i(x) = Cp_i(x)$
  - f_i is weakly relevant given B and E.

Fig. 6. A partial specification for an ILP system for feature construction.

3.2 Implementation

Here we describe an implementation that can be tailored to meet the specification just described. Recall that all the specification, with our subsequent modifications, requires is the following: (1) The implementation returns a set in which each feature is represented as a conjunction of literals; (2) Each feature is constructed from a mixed definite clause that entails at least one example, given the background knowledge; and (3) Each feature is weakly relevant, given the background knowledge and

[6] In [31] the f_i are denoted by $f_{i,c}$. The notation here is more in line with the machine learning literature.

Clause:

$$h_1 : \quad Class(x, voltar) \leftarrow Has\_expression(x, come\_back, voltar) \\ Has\_pos(x, pcwr\_4, nn)$$

Context Predicate:

$$Cp_1(x) : \quad Has\_expression(x, come\_back, voltar) \wedge Has\_pos(x, pcwr\_4, nn)$$

Feature:

$$f_1(x) = \begin{cases} 1 \; Has\_expression(x, come_back, voltar) \wedge Has\_pos(x, pcwr\_4, n) \; = \; 1 \\ 0 \text{ otherwise} \end{cases}$$

Fig. 7. Example of a boolean feature constructed from a clause for WSD. The clause shown here identifies the Portuguese sense of the English verb "to come". The meanings of the predicate symbols $Has\_expression$ and Has_pos are explained in Section 4.

examples. Clearly a very large number of features can be constructed that satisfy these requirements. Consequently, we introduce the following general constraints for restricting the number of features constructed:

Support. For each clause $h_i : Class(x, c) \leftarrow Cp_i(x)$ we compute $P(h_i) = \{x : e \in E \wedge e = Class(x, c) \wedge B \cup \{h_i\} \cup \{\neg e\} \models \square\}$. $P(h_i)$ is the set of individuals correctly classified by h_i, and we denote their number by $tp(h_i) = |P(h_i)|$ (more correctly of course, we should include B and E in these functions). Adopting data-mining terminology, we call this the "support" of a clause. In our implementation, we reduce the number acceptable clauses found by the ILP system (and hence the number of features) by placing some minimal requirement on the support of any clause.

Precision. For each class $c \in \mathcal{Y}$, we denote $N_c = \{\neg Class(x, c) : Class(x, c') \in E \wedge c \neq c'\}$. Then for each clause $h_i : Class(x, c) \leftarrow Cp_i(x)$ we compute $N(h_i) = \{x : e \in N_c \wedge e = \neg Class(x, c) \wedge B \cup \{h_i\} \cup \{e\} \models \square\}$. $N(h_i)$ is the set of individuals incorrectly classified by a h_i as class c, and we denote their number by $fp(h_i) = |N(h_i)|$. Once again, using terminology from the data-mining literature, we obtain the precision of a clause h_i as the value $tp(h_i)/(tp(h_i) + fp(h_i))$. In our implementation, we reduce the number acceptable clauses found by the ILP system (and hence the number of features) by requiring the precision of any clause to be greater than some pre-specified number.

We note that ensuring that the support of a clause is at least 1 meets the requirement that the clause entails at least one example. The constraint on precision can be used to ensure that features constructed are weakly relevant. This follows trivially, since each clause h for class c results in a feature f and the following contingency table for class values:

		Class c	Class $\overline{c}$	
f	1	$tp(h)$	$fp(h)$	n_a
	0	$N_c - tp(h)$	$N_{\overline{c}} - fp(h)$	n_b
		N_c	$N_{\overline{c}}$	N

If the precision constraint ensures that the precision of a clause h for a class c is greater than the prior probability of class c (this probability is estimated from the examples E), then $precision(h) = tp(h)/n_a > N_c/N = Pr(Class = c)$. Now, since $Pr(Class = c|f = 1) = precision(h)$, it follows that $Pr(Class|f) \neq Pr(Class)$. That is, the corresponding feature f is weakly relevant.

We will call any clauses (and, with some overloading of terminology, the corresponding features) that satisfies the support and precision requirements as "acceptable". We distinguish between two different techniques to obtain features:

1. One-shot feature-set construction, that identifies a set of acceptable features without any feedback from the model constructor (this is, in some sense, a "LINUS-inspired" approach [14]). In keeping with terminology sometimes used in the ILP literature, we will call this approach "static feature-set construction".
2. Incremental feature-set construction, that identifies a set of acceptable features after repeated iterations in which feature-construction is guided by feedback from the model constructor. In keeping with terminology sometimes used in the ILP literature, we will call this approach "dynamic feature-set construction".

We present these techniques as instances of the implementation below.

Randomised Feature-Set Construction

In [44] an implementation using a randomised local search was proposed to meet the specification of a predictive ILP system. It is possible to adopt the same approach in an implementation designed to meet the specifications for feature construction (see Fig. 8).

In Fig. 8 R and M bound the number of random restarts and local moves allowed. Existing techniques for feature construction can be recast as special cases of this procedure, with appropriate values assigned to R and M; and definitions of a starting point (Step 3a) and local moves (Step 3(f)i). For example, one-shot methods that use an ILP engine to construct a large number of features independent of the model constructor can be seen as an instance of the randomised procedure with $R = 1$ and $M = 0$, with some technique for generating the "starting subset"

```
1. bestfeatures:= {}
2. bestaccuracy:= 0.0
3. for i = 1 to R do begin
   (a) currentfeatures:= randomly selected set of features
   (b) currentmodel:= model constructed with currentfeatures
   (c) accuracy:= estimated accuracy of currentmodel
   (d) if accuracy > bestaccuracy then begin
          i. bestfeatures:= currentfeatures
         ii. bestaccuracy:= accuracy
   (e) end
   (f) for j = 1 to M do begin
          i. nextfeatures:= best local move from currentfeatures
         ii. nextmodel:= model constructed with nextfeatures
        iii. accuracy:= estimated accuracy of nextmodel
        iv. if accuracy > bestaccuracy then begin
              A. bestfeatures:= nextfeatures
              B. bestaccuracy:= accuracy
         v. end
        vi. currentfeatures:= nextfeatures
   (g) end
4. end
5. return bestfeatures
```

Fig. 8. A basic randomised local search procedure, adapted to the task of feature construction.

in Step 3a. The feature-set constructor provided within the Aleph [39] system, shown in Fig. 9(a), is an example of a technique that generates features using a stratified sample of the examples (a simpler approach of generating k acceptable features using simple random sampling of the examples is in Fig. 9(b)).

We consider now the more general procedure shown. In this, R and M can take on any value from the set of natural numbers (including 0). In addition, the starting subset is assumed to be drawn using some distribution that need not be known; and a local move is one that either adds a single new feature to the existing subset of features, or drops a feature from the existing subset. We are now immediately confronted with two issues that make it impractical to use the procedure as shown. First, we have the same difficulty that prevented us from using an enumerative technique like a Gibb's sampler: generating the local neighbourhood requires us to obtain all possible single-feature additions. Second, for each local move, we need to construct a model, which can often be computationally expensive. We address each of these in turn.

Reducing Local Moves. We consider a modification of the search procedure in Fig. 8 that results in only examining a small sample of all the local moves possible before deciding on the next move in Step 3(f)i. Ideally, we are interested in obtaining a sample that, with high probability, contains the best local move possible. Assuming there are no ties, and that the number of possible local moves is very large, it would clearly be undesirable to select the sample using a uniform distribution over local moves. We propose instead a selection that uses the errors made by the model-constructor to obtain a sample of local moves. As a result, features in the local neighbourhood that are relevant to the errors are more likely to be selected. In some sense, this is somewhat reminiscent of boosting methods: here, instead of increasing the weights of examples incorrectly classified, the representation language is enriched in a way that is biased

$strat\_features(B, E, \mathcal{Y}, k, s, p, \mathcal{L})$: Given background knowledge B; a set of examples E; a set of class-labels $\mathcal{Y}$ taken here to be the set of numbers $\{1, 2, \ldots, c\}$; an upper-bound on the number of features allowed for each class k ($1 \leq k$); the minimal support for acceptable clauses s ($1 \leq s$); the minimal precision for acceptable clauses p ($0 < p \leq 1$); and a language $\mathcal{L}$ specifying constraints on acceptable clauses that can be considered in the search for any one feature; return a set F of at most k acceptable features for each class.

1. for each $i \in \mathcal{Y}$ do
2. begin
   (a) Let $H_i := \emptyset$
   (b) Let E_i be all examples in E with class i
   (c) while ($|H_i| \leq k$) and ($E_i \neq \emptyset$) do
   (d) begin
      i. Randomly select an example $e \in E_i$
      ii. Let $j := k - |H_i|$
      iii. Let H be a set of at most j mixed definite clauses found by an ILP engine such that $H \cap H_i = \emptyset$, and each clause $h \in H$ satisfies: (1) $h \in \mathcal{L}$; (2) $B \cup \{h\} \models e$; and (3) h is acceptable (that is, $support(h) \geq s$; and $precision(h) \geq p$).
      iv. Let $H_i := H_i \cup H$
      v. Let $E_i := E_i - \{e\}$
   (e) end
3. end
4. Let F be the set of features obtained from the bodies of clauses in $H_1 \cup H_2 \cdots H_c$
5. return F

(a) A procedure for constructing a stratified sample of acceptable features.

$srs\_features(B, E, k, s, p, \mathcal{L})$: Given background knowledge B; a set of examples E; a set of class-labels $\mathcal{Y}$ taken here to be the set of numbers $\{1, 2, \ldots, c\}$; an upper-bound on the number of features allowed k ($1 \leq k$); the minimal support for acceptable clauses s ($1 \leq s$); the minimal precision for acceptable clauses p ($0 < p \leq 1$); and a language $\mathcal{L}$ specifying constraints on acceptable clauses that can be considered in the search for any one feature; return a set F of at most k acceptable features.

1. Let $H = \emptyset$
2. Let E' be a random subset of k examples from E
3. while ($|H| \leq k$) and ($E' \neq \emptyset$) do
4. begin
   (a) Randomly select an example $e \in E'$
   (b) Let $j := k - |H|$
   (c) Let H' be a set of at most j mixed definite clauses found by an ILP engine such that $H \cap H' = \emptyset$, and each clause $h \in H'$ satisfies: (1) $h \in \mathcal{L}$; (2) $B \cup \{h\} \models e$; and (3) h is acceptable (that is, $support(h) \geq s$; and $precision(h) \geq p$).
   (d) Let $H := H \cup H'$
   (e) Let $E' := E' - \{e\}$
5. end
6. Let F be the set of features obtained from the bodies of clauses in H
7. return F

(b) A procedure for constructing a simple random sample of acceptable features.

Fig. 9. Candidates for using an ILP engine to generate the "starting subset" in the randomised search for features. *support* and *precision* are as described in the text.

to classify these examples correctly on subsequent iterations (the possibility of using misclassified examples to guide a stochastic local search amongst an existing set of features was demonstrated in [25]).

Recall that at any point, a local move from a feature-subset F is obtained by either dropping an existing feature in F or adding a new feature to F. We are specifically concerned with the addition step, since in principle, all possible features that can be constructed by the ILP engine could be considered as candidates. We curtail this in the following ways. First, we restrict ourselves to samples of features that are related to examples misclassifed by the model-constructor using the current set of features F (by "related" we mean those that are TRUE for at least one of the examples in error). Second, in our implementation, we restrict ourselves to a single new feature for each such example (by "new", we mean a feature not already in F).[7]

Reducing models constructed. The principal purpose of constructing and evaluating models in the local neighbourhood is to decide on the best next move to make. This will necessarily involve either an addition of a new feature to the existing set of features, or the deletion of an existing feature from the current set of features. That is, we are looking to find the best new feature to add, or the worst old feature to drop (given the other features in the set, of course). Correctly, we would form models with each old feature omitted in turn from the current set and each new feature added in turn to the current set. The best model would then determine the next move. Using a model-constructor that assigns weights to features allows us to adopt the following sub-optimal procedure instead. First, we find the feature with the lowest weight in the current model: this is taken to be the worst old feature. Next, we construct a single model with all features (old and new). Let us call this the "extended model". The best new feature is taken to be the new feature with the highest weight in the extended model

The procedure in Fig. 8 with these modifications is shown in Fig. 10. It is evident that the number of additional models constructed at any point in the search space is now reduced to just 3: the price we pay is that we are not guaranteed to obtain the same result as actually performing the individual additions and deletions of features.

It is the randomised local search procedure in Fig. 10, with one small difference, that we implement in this paper. The difference arises from the comparison of models: in the procedure shown, this is always done using estimated accuracies only. In our implementation, if estimated accuracies for a pair of models are identical, then the model using fewer features is prefered (that is, comparisons are done on the pair (A, F) where A is the estimated accuracy of the model and F is the number of features used in the model). One additional point worth clarifying concerns over-fitting the data. Some combination of the following mechanisms are clearly possible: (a) Limiting the amount of local search (using M); (b) Requiring acceptable features have some reasonable support; and (c) Us-

[7] In the implementation, we select this feature using its discriminatory power given the original set of examples.

$randsearch\_features(B, E, \mathcal{Y}, n, s, p\mathcal{L}, R, M)$: Given background knowledge B; a set of examples E; a set of class-labels $\mathcal{Y}$ taken here to be the set of numbers $\{1, 2, \ldots, c\}$; a sample size n $(1 \leq n)$; the minimal support for acceptable clauses s $(1 \leq s)$; the minimal precison for acceptable clauses p $(0 < p \leq 1)$; and a language $\mathcal{L}$ specifying constraints on acceptable clauses that can be considered in the search for any one feature; the number of random restarts R $(1 \leq R)$; and the number of local moves M $(1 \leq M)$, return a set of acceptable features F.

1. F:=
2. bestaccuracy:= 0.0
3. for i = 1 to R do begin
   (a) currentfeatures:= sample of acceptable features given $B, E, \mathcal{Y}, n, s, p\mathcal{L}$
   (b) currentmodel:= model constructed with currentfeatures
   (c) accuracy:= estimated accuracy of currentmodel
   (d) if accuracy > bestaccuracy then begin
      i. F:= currentfeatures
      ii. bestaccuracy:= accuracy
   (e) end
   (f) for j = 1 to M do begin
      i. F_{new}:= sample of new acceptable features that are TRUE for errors made by currentmodel
      ii. extendedmodel:= model constructed using currentfeatures and F_{new}
      iii. f_{worst}:= feature in currentfeatures with lowest weight in currentmodel
      iv. f_{best}:= feature in F_{new} with highest weight in extendedmodel
      v. F^-:= set with feature subset obtained by dropping f_{worst} from currentfeatures
      vi. F^+:= set with feature subset obtained by adding f_{best} to currentfeatures
      vii. localmoves:= $F^- \cup F^+$
      viii. nextfeatures:= best subset in localmoves
      ix. nextmodel:= model constructed with nextfeatures
      x. accuracy:= estimated accuracy of nextmodel
      xi. if accuracy > bestaccuracy then begin
         A. F:= nextfeatures
         B. bestaccuracy:= accuracy
      xii. end
      xiii. currentfeatures:= nextfeatures
      xiv. currentmodel:= nextmodel
   (g) end
4. end
5. return F

Fig. 10. The randomised local search procedure for feature construction, modified using theory-guided sampling of local moves and the use of feature-weights to reduce model construction. Acceptable features are obtained using the same technique as the static feature-set constructor: an example for which the feature is required to be $TRUE$ is selected, and then an acceptable clause found.

ing a model constructor that can perform some appropriate trade-off of fit-versus-complexity to avoid over-fitting.

We make the following observations about this implementation:

Termination. It is evident that, provided the ILP engine used construct features terminates, the procedure in in Fig. 10 terminates, since both R and M are finite.

Correctness. All features constructed are obtained from acceptable clauses. That is, the support of every clause is at least s $(s > 1)$, and, precision is at least p. If p is ensured to be greater than the prior probablity of any class c in the examples E, then from the discussion earlier, the corresponding features are all weakly relevant.

Incompleteness. The procedure does not identify all weakly relevant features, since the local search procedure is incomplete.

Output. We assume that the starting subset in Step 3a contains features proportional to n (for example, with c classes, a starting subset using the Aleph feature-set constructor in Fig 9 will produce a set containing no more than cn features). On any restart, a local move

either adds a single feature to the existing set of features; or drops a feature from this set. Since this done no more than M times, the output contains more than $\alpha n \pm M$ features ($1 \leq \alpha < \infty$).

Based on the observations that: (a) the procedure in Fig. 10 terminates; and (b) with a value of p that can be determined from the examples, it returns a finite set of weakly relevant features, each constructed from clauses that entail at least 1 example, we can view it as an algorithm for feature construction.

4 Empirical Evaluation

Our objective is to evaluate empirically if ILP-based features can assist in constructing models for WSD. Specifically, given different sources of background information, we intend to investigate the performance of the following kinds of models:

Baseline models. These are models for WSD that use features that are obtained manually from the different knowledge sources.

ILP-assisted models. These are models constructed by augmenting the hand-crafted features with those found by either the static or dynamic feature-set construction techniques, when provided with access to a richer representation of the same background information as the Baseline models. When static feature-set construction is used, we denote the resulting ILP-assisted model "ILP-S"; and when dynamic feature-set construction is used, we denote the corresponding model "ILP-D".

We wish to investigate whether the ILP-assisted models have a significantly higher predictive accuracy than the baseline models. Statistical evidence in favour of the ILP-assisted models will, in turn, be taken as providing support for the "ILP-features hypothesis", which we reiterate here:

> **ILP-features hypothesis.** Feature-construction by ILP provides a general method of introducing background knowledge for improving the predictive accuracy of WSD models.

Statistically speaking, the null hypothesis for the investigation is there is no difference in the (average) predictive accuracies of Baseline and ILP-S or ILP-D models. Inability to reject this hypothesis will be taken as evidence against the usefulness of ILP as a feature-constructor for WSD.

4.1 Materials

Data We experiment with datasets contained in three different tasks. Each collection is concerned with the disambiguation of different sets of ambiguous words, the so called "target words", including nouns and

verbs, and contain different numbers of classes (possible senses or translations) and class distributions. The three tasks are:

Monolingual (1). Data consist of the 32 verbs from the SENSEVAL-3 competition. SENSEVAL[8] is a joint evaluation effort for WSD and related tasks. We use all the verbs of the English lexical sample task from the third edition of the competition. The number of examples for each verb varies from 40 to 398 (average of 186). The number of senses varies from 3 to 12 with an average of 7 senses. The average accuracy of the majority class is about 55%. We refer the reader to [18] for more information about the SENSEVAL-3 data.

Monolingual(2). Data consist of 85 verbs and nouns of the English lexical sample task from the SemEval-2007 competition, the last edition of SENSEVAL. The number of examples varies from 24 to 3,061 (average of 272.12). The number of senses used in the training examples for a given word varies from 1 to 13 (average of 3.6). The average accuracy of the majority class is about 78%. More information about this data set can be found in [28].

Bilingual. Data consist of 7 highly frequent and ambiguous, mostly content-light, verbs: *come*, *get*, *give*, *go*, *look*, *make*, *take*. The sample corpus comprises around 200 English sentences for each verb extracted from several corpora, including the European Parliament, the Bible and fiction books, with the verb translation automatically annotated [38]. The number of possible translations varies in the corpus from 5 to 17, with an average of 11 translations. The average accuracy of the majority class in the test data is about 54%.

Taken together, the tasks specify 124 independent datasets, with disambiguation required ranging from 3 to 17 different senses. The total number of examples is approximately $27,000$.

Background Knowledge To achieve accurate disambiguation in both tasks is believed to require a variety of lexical, syntactic and semantic information. In what follows, we describe the background knowledge available for the tasks and illustrate it using the following sentence (assuming that we are attempting to determine the sense of 'coming' as target word):

> "If there is such a thing as reincarnation, I would not mind coming back as a squirrel."

Background knowledge is available in the following categories:

B0. *Baseline features*. Manually identified features using the information encoded in the predicates below, conveying the same information, but represented by means of attribute-value vectors.

[8] http://www.senseval.org

B1. *Bag-of-words.* The 5 words to the right and left of the target word, extracted from the corpus and represented using definitions of the form $Has\_bag(sentence, word)$. For example:

$$Has\_bag(snt1, mind).$$
$$Has\_bag(snt1, not). \ldots$$

B2. *Narrow context.* Lemmas of 5 content words to the right and left of the target word, extracted from the corpus, previously lemmatized by MINIPAR [16]. These are represented using definitions of the form $Has\_narrow(sentence, wordposition, word)$. For example:

$$Has\_narrow(snt1, first\_content\_word\_left, mind).$$
$$Has\_narrow(snt1, first\_content\_word\_right, back). \ldots$$

These are not provided for the Bilingual task, since it was thought to be adeqately covered by B5 below.

B3. *Part-of-speech tags.* Part-of-speech (POS) tags of 5 content words to the right and left of the target word, are obtained using MXPOST [31] and represented using definitions of the form: $Has\_pos(sentence, wordposition, pos)$. For example:

$$Has\_pos(snt1, first\_content\_word\_left, nn).$$
$$Has\_pos(snt1, first\_content\_word\_right, rb). \ldots$$

B4. *Subject-Object relations.* Subject and object syntactic relations with respect to the target word - in case it is a verb. If it is a noun, the representation includes the verb of which it is a subject or object, and the verb / noun it modifies.These were obtained from parsing sentences using MINIPAR and represented using definitions of the form $Has\_rel(sentence, type, word)$. For example:

$$Has\_rel(snt1, subject, i).$$
$$Has\_rel(snt1, object, nil). \ldots$$

B5. *Word collocations.* 11 collocations with respect to the target word, extracted from the corpus: 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. These are represented using definitions of the form $Has\_collocation(sentence, collocation\_type, collocation)$. For example:

$$Has\_collocation(snt1, first\_word\_right, back).$$
$$Has\_collocation(snt1, first\_word\_left, mind). \ldots$$

B6. *Verb restrictions.* Selectional restrictions of the verbs, when these are the target words, defined in terms of the semantic features of their arguments in the sentence, extracted using LDOCE [29]. If the restrictions imposed by the verb are not part of the description of its arguments, WordNet relations are used to check whether they can be satisfied by synonyms or hyperonyms of those arguments. A hierarchy of feature types is used to account for restrictions established by the verb that are more generic than the features describing its arguments in the sentence. These are represented by definitions of the form $Satisfy\_restrictions(sentence, rest\_subject, rest\_object)$. For example:

$$Satisfy\_restrictions(snt1, [human], nil).$$
$$Satisfy\_restrictions(snt1, [animal, human], nil).$$

B7. *Dictionary definitions.* A relative count of the overlapping words in dictionary definitions of each of the possible senses of the target word (extracted from [26], for the bilingual task, and from [29], for the monolingual tasks) and the words surrounding it in the sentence, to find the sense with the highest number of overlapping words. These are represented by facts of the form $Has\_overlap(sentence, translation)$. For example:

$$Has\_overlap(snt1, voltar).$$

B8. *Phrasal verbs.* Phrasal verbs involving the target verb, possibly occurring in a sentence, according to the list of phrasal verbs given by dictionaries and the context of the verb (5 surrounding words). These are represented by definitions of the form $Has\_expression(sentence, verbal\_expression)$. For example:

$$Has\_expression(snt1, \text{'come back'}).$$

This is not provided for Monolingual task (1) since it does not consider senses of the verbs occurring in phrasal verbs.

B9. *Frequent bigrams.* Bigrams consisting of pairs of adjacent words in a sentence (without the target word) which occur more than 10 times in the corpus and are represented by definitions of the form $Has\_bigram(sentence, word_1, word_2)$. For example:

$$Has\_bigram(snt1, such, a).$$
$$Has\_bigram(snt1, back, as). \ldots$$

These are available for Monolingual task (2) only.

B10. *Frequently related words.* Related words consisting of pairs in the sentence that occur in the corpus more than 10 times related by verb-subject, verb-object, verb-modifier, subject-modifier, and object-modifier syntactic relations, without including the word to be disambiguated. There are represented by facts of the type $Has\_related\_pair(sentence, word_1, word_2)$. For example:

$$Has\_related\_pair(snt1, there, is). \ldots$$

These are available for Monolingual task (2) only.

Of these definitions, B0 is intended for constructing the Baseline model. B1–B10 are intended for use by an ILP system. For each task we used a different subset of these knowledge sources, according to their availability. Further, although the ILP implementation we use is entirely capable of exploring intensional definitions of each of B1–B10, we represent definitions in an extensional form (that is, as a set of ground facts), since this representation is more efficient. This results in about 1.4 million facts.

Algorithms We distinguish here between three separate procedures: (1) The feature-construction algorithm in Fig. 10 for constructing a set of acceptable features; (2) The ILP engine concerned with identifying clauses required by the feature-set construction algorithms; and (3) The procedure for constructing models given a set of features and their values for sample instances, along with a class label associated with each instance. We have an implementation, in the Prolog language, of Fig. 10. For static-feature construction, the stratified sampling procedure provided within the Aleph system shown in Fig. 9(a) is used to provide the "starting subset" of features. In other cases, we commence with a simple

random sample of features using the procedure in Fig. 9(b). For (2) we use a basic branch-and-bound search procedure provided within Aleph, that attempts to find all clauses that satisfy a given set of constraints and entail at least one example. The feature-based model constructor (3) is a linear SVM (the specific implementation used is the one provided in the WEKA toolbox called SMO.[9]). We will refer to (1) as "the feature constructor"; (2) as "the ILP learner" and (3) as "the feature-based learner."

4.2 Method

Our method is straightforward:

For each verb in each task (that is, 32 words in Monolingual(1), 85 words in Monolingual(2) and 7 words in the Bilingual task):

1. Obtain the best model possible using the feature-based learner and the features in B0. Call this the Baseline model.
2. Construct a set of features using, in turn, the static and dynamic feature construction that uses the ILP learner, equipped with background knowledge definitions B1–B10. Call the corresponding sets of features "Static" and "Dynamic".
3. Obtain the best model possible using the feature-based learner supplied with data for features B0 ∪ Static. Call this model "ILP-S". Similarly, obtain "ILP-D" using features B0 ∪ Dynamic. For simplicity, we will refer to these models collectively as "ILP-assisted models".
4. Compare the performance of the Baseline model against that of the ILP-assisted models.

The following details are relevant:

(a) For the monolingual tasks, we use the training/test data as provided by SENSEVAL-3 and SemEval-2007 benchmarks. These specify different percentages for training and test, depending on the target word. For the bilingual task, we use 25% of the data to estimate the performance of disambiguation models. For all the data sets, performance is measured by the accuracy of prediction on the test set (that is, the percentage of test examples whose sense is predicted correctly).

(b) The ILP learner constructs a set of acceptable clauses in line with the specifications described in Section 3.1. Positive examples for the ILP learner are provided by the correct sense (or translation in the bilingual case) of the target word in a sentence. Negative examples are generated automatically using all other senses (or translations). Clauses for a class are thus found using a "one-versus-the-rest" approach.

(c) For each target word and task, constructing the "best possible model" requires determining optimal values for some parameters of the procedures involved. We estimate these values using an instance of the method proposed in [10] that proceeds as follows. First, we decide

[9] http://www.cs.waikato.ac.nz/~ml/weka/

on the relevant parameters. Second, we obtain, using the training set only, unbiased estimates of the predictive accuracy of the models for each target word arising from systematic variation across some small number of values for these parameters. Values that yielded the best average predictive accuracy across all target words are taken to be optimal ones. This procedure is not perfect: correctly, optimal values may change from one target word to another; and even if they did not, the results obtained may be a local maximum (that is, better models may result from further informed variation of values).

(d) Full-scale experimentation for optimal settings for parameters requires systematic joint variation of values of critical parameters for the feature constructor, ILP learner and the feature-based learner. For reasons of tractablility, we restrict this here to an exploration of parameter values for the feature-based learner (here, a linear SVM) only. For the other two procedures, we select values that allow reasonably large numbers of features to be generated (see below). The principal parameters for the feature-based learner are taken to be: the C parameter used by the linear SVM and F, the number of features to be selected. The following C values were investigated: $0.1, 1.0, 10, 100$, and 1000. Given a total of N features, F values considered were $N/64, N/32, N/16, N/8, N/4, N/2$, and N (we obtain the best subset of such features using software tools for feature-selection provided within WEKA). The predictive accuracy with each (C, F) setting is estimated and the values that yield the best results are used to construct the final model (the predictive accuracy estimate is obtained using an average over 5 repeats of predictions on 20% of the training data sampled to form a "validation" set).

(e) For the record, parameter settings for the feature-constructor and ILP learner were as follows. For the former the value of k (the maximum number of features constructed for each class) is 5000; the minimal support s required is 2; and the minimal precision 0.6. We note that this value of precision does always guarantee weak relevance in all datasets (where the prior probability of a class is > 0.6). In such cases, not all the features may be weakly relevant. In addition, the search is bounded by requiring clauses to have no more than 10 literals in the context-predicate (that is, the body of clauses has at most 10 literals). Search for clauses is restricted to no more than $10,000$ nodes. All these settings are admittedly *ad hoc*, with the intent being to place, within computational reason, as few constraints as possible on feature generation.

(f) Comparison of performance is done using the Wilcoxon signed-rank test [33]. This is a non-parametric test of the null hypothesis that there is no significant difference between the median performance of a pair of algorithms. The test works by ranking the absolute value of the differences observed in performance of the pair of algorithms. Ties are discarded and the ranks are then given signs depending on whether the performance of the first algorithm is higher or lower than that of the second. If the null hypothesis holds, the sum of the signed ranks should be approximately 0. The probabilities of observing the actual signed rank sum can be obtained by an exact

calculation (if the number of entries is less than 10), or by using a normal approximation. We note that the comparing a pair of algorithms using the Wilcoxon test is equivalent to determining if the area under the ROC curves of the algorithms differ significantly [**?**].

(g) We note also that given that the ILP-assisted models are constructed with access to all features available for the Baseline model, we would expect that the predictive accuracies of the ILP-assisted models should, in principle, never be worse. That is, the alternate hypothesis for the Wilcoxon test is a uni-directional one that the median accuracy of the ILP-assisted model (ILP-S or ILP-D) is higher than the Baseline model. However, in practice, ILP-assisted models could be worse either due to overfitting, or due to limitations of the feature-selection procedure employed (if any). Given this, we adopt the bi-directional alternate hypothesis, that the median accuracies of the ILP-assisted models are not equal to the Baseline model. This means obtaining two-tailed probabilities for the signed rank-differences.

5 Results and Discussion

Figures 14, 15 and 16 in Appendix 6 tabulate the performance of the Baseline and ILP-assisted models on the three different WSD tasks. It is also standard practice to include the performance of a classifier that simply predicts the most frequent sense of the target word, which we denote in these tabulations as the "Majority class" model. The principal details in these tabulations are these: (1) The majority class classifier clearly performs poorest, and we will leave them out of any further discussion; (2) For all three tasks, average accuracies of the baseline models are lower than the ILP-assisted models.

We turn now to the question of whether the differences in accuracies observed between the Baseline and ILP-assisted models are in fact significant. The relevant probabilities calculated by using the Wilcoxon test are shown in Fig. 11.[10] The tabulations show that, overall (see Fig. 11(a)), there is very little evidence in favour of the null hypothesis that the median accuracies of the ILP-assisted models are the same as the Baseline model. Closer examination of the individual tasks (Fig. 11(b)–(d)) suggests that we can be quite confident about this with ILP-D. ILP-S is also better, but some of the probabilities on Monolingual (1) and Bilingual tasks are slightly lower than what would usually be considered significant.

These tabulations suggest that there is indeed evidence in favour of the ILP-features conjecture, using ILP-S or ILP-D as an example of what is achievable with ILP assistance. One question of relevance to this conclusion is this: how do we know that the Baseline models are not deliberately poor? Clearly, we are not able to answer this question decisively. Nevertheless, setting aside the possibility of obtaining better models by using a different feature-based learner, we provide some evidence that

[10] These were obtained from the program kindly provided by Richard Lowry at `http://faculty.vassar.edu/lowry/wilcoxon.html`.

the hand-crafted features, although not provably the best possible, are nevertheless very good. A comparison of the Baseline models against the state-of-the-art models reported for the Monolingual(1) dataset, shown in Fig. 12 below, suggests that these features yield models that are comparable to the best available (without ILP assistance).[11]

	Base	ILP-S	ILP-D
Base	–	–	–
ILP-S	< 0.0001	–	–
ILP-D	< 0.0001	< 0.0001	–

(a) Overall

	Base	ILP-S	ILP-D
Base	–	–	–
ILP-S	0.06	–	–
ILP-D	0.0006	0.002	–

(b) Monolingual (1)

	Base	ILP-S	ILP-D
Base	–	–	–
ILP-S	0.0001	–	–
ILP-D	< 0.0001	0.002	–

(c) Monolingual (2)

	Base	ILP-S	ILP-D
Base	–	–	–
ILP-S	0.08	–	–
ILP-D	0.02	> 0.10	–

(d) Bilingual

Fig. 11. Probablities of observing the differences in accuracies for the monolingual and bilingual tasks, under the null hypothesis that median accuracies of the pair of algorithms being compared are equal. Each entry consists two-tailed probability estimates of the null hypothesis being true.

Finally, although not relevant to the principal hypothesis being investigated in the paper, but nevertheless of some interest for both automatic WSD and the ILP practitioner is the difference between static and dynamic feature construction techniques embodied within ILP-S and ILP-D. On the evidence in the Appendix of this paper and Fig. 11, it appears that, for WSD tasks at least, ILP-D is better (although results are not significantly better on the Bilingual subset). Besides their better predictive performance, ILP-D models also require substantially fewer features—this is after the feature selection step—than ILP-S (and sometimes even the baseline models: see Fig. 13), arising from the use of a partially correct model to direct the construction of only those features that may be necessary to correct the model.

6 Concluding Remarks

Word sense disambiguation, a necessary component for a variety of natural language processing tasks, remains amongst the hardest to model adequately. It is of course possible that the vagaries of natural language may

[11] None of this, of course, address the question of whether features capturing some or all of the information found by the ILP learner *could* be hand-crafted. This is beyond the scope of this paper.

Models	**Accuracy**
MC-WSD [4]	72.50
Baseline	69.94
Syntalex-3 [20]	67.60
Syntalex-1 [20]	67.00
CLaC1 [12]	67.00
Syntalex-2 [20]	66.50
CLaC2 [12]	66.00
Syntalex-4 [20]	65.30
Majority class	55.31

Fig. 12. A comparison of the Baseline model against the best submissions made for the SENSEVAL-3 competition, that is, in the Monolingual (1) dataset. The best model, MC-WSD, is a multi-class averaged perceptron classifier with one component trained on the data provided by SENSEVAL and on WordNet glosses.

Model	**Task**		
	Monolingual (1)	Monolingual (2)	Bilingual
Baseline	136	282	102
ILP-S	2429	2813	2918
ILP-D	179	133	63

Fig. 13. Average numbers of features used by the different models.

place a limit on the accuracy with which a model could identify correctly the sense of an ambiguous word, but it is not clear that this limit has been reached with the modelling techniques that constitute the current state-of-the-art. The performance of these techniques depends largely on the adequacy of the features used to represent the problem. As it stands, these features are usually hand-crafted and largely of a lexical nature. For substantial, scalable progress it is believed that knowledge that accounts for more elaborate syntactic and semantic information needs to be incorporated. In this paper, we have investigated the use of Inductive Logic Programming as a mechanism for incorporating multiple sources of syntactic and semantic information into the construction of models for WSD. The investigation has been in the form of empirical studies of using ILP to construct features that are then used to construct predictive models for monolingual and bilingual WSD tasks and the results provide evidence that this approach can yield better models.

We believe much of the gains observed with ILP stems from the use of substantial amounts of background knowledge. For the work here, this

knowledge has been obtained by translations of information in standard corpora or electronic lexical resources. This is promising, as it suggests that these translators, in conjunction with ILP, may provide a set of tools for the automatic incorporation of deep knowledge into the construction of general WSD models. Turning specifically to the findings of this paper, a combination of randomised search and incremental feature-set construction by an ILP engine appears to be a promising method to construct good features for WSD.

There are a number of ways in which the work here can be improved and extended. We list the main limitations here under three categories. On the conceptual front, it is evident that we have not provided any guarantees of optimality on the feature-subset constructed. While this is typical of randomised methods of the type proposed here, it would nevertheless be useful to obtain some performance bounds, however loose. Our specification of a feature-construction algorithm—to the best of our knowledge, the only one to be proposed in the ILP literature—can be refined to require the algorithm to return strongly relevant features (in the sense described by [9]).

On the implementation front, our implementation of dynamic feature-set construction is based on the simplest kind of randomised search (GSAT). Better methods exist and need to be investigated (for example, WalkSat). Further, we could consider other neighbourhood definitions for the local search such as adding or dropping upto k features. Of course, we are not restricted to use SVMs, or even the specific variant of SVM here, as our model constructor. Our experiments on the monolingual tasks here suggest that there is no significant difference between a "1-norm" SVM and the standard approach we have used here, but other model construction techniques may yield better results. We note that the feature-set constructor when used in conjunction with an SVM is an instance of SVILP, as proposed in [23]. More generally, by interleaving feature and statistical model construction in the dynamic feature-set constructor, we are effectively performing a form of statistical relational learning. The applicability of this approach to this wider area needs to be investigated.

On the application front, the background knowledge used here is by no means exhaustive of the kind available for the WSD tasks studied here. For example, for the bilingual task, the "translation context" for a target word may help greatly. This refers to the translations into the target language of the words forming the context of the target word. Our preliminary experiments show that this does improve accuracies on the bilingual task. Further, the algorithms we have described here are unlikely to be the only ones to satisfy the specifications for feature-constructors. While the procedures in the paper establish a case for ILP-based feature construction in WSD, other feature-construction methods such as the ones in [13] and [6] may yield even better models, thus strengthening the case for ILP further.

On the flip side, it is evident that the feature-construction algorithm we have proposed here are not specific to WSD data. The use of the algorithm to model data concerned with tasks other than WSD, while outside the scope of this paper, has obvious wider interest.

Acknowledgements

This work was begun when Lucia Specia visited Ashwin Srinivasan at the IBM Research Laboratory in New Delhi. Jimmy Foulds, at the University of Waikato, helped beyond the call of duty in getting an executable version of code that constructed models using a 1-norm SVM within WEKA.

References

1. Agirre, E. and Rigau, G.: Word Sense Disambiguation Using Conceptual Density. 16th International Conference on Computational Linguistics, Copenhagen 16–22 (1996)
2. Bar-Hillel, Y. Automatic Translation of Languages. In F. Alt, D. Booth, and R. E. Meagher (eds), Advances in Computers. Academic Press, New York (1960)
3. Cannon, E.O., Amini, A., Bender, A., Sternberg, M.J.E., Muggleton S.H., Glen, R.C., Mitchell, J.B.O. Support vector inductive logic programming outperforms the naive Bayes classifier and inductive logic programming for the classification of bioactive chemical compounds. J Comput Aided Mol Des, 21:269–280 (2007)
4. Ciaramita, M. and Johnson, M. Multi-component Word Sense Disambiguation. SENSEVAL-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona 97–100 (2004)
5. Cottrell, G. W. A Connectionist Approach to Word Sense Disambiguation. Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo (1989)
6. Davis, J., Ong, I., Struyf, J., Burnside, E., Page, D., and Costa, V. S.: Change of representation for statistical relational learning, International Joint Conferences on Artificial Intelligence, (2007) Hand, D.J.: *Construction and Assessment of Classification Rules*, John Wiley, Chichester (1997).
7. Hirst, G. Semantic Intepretation and the Resolution of Ambiguity. Studies in Natural Language Processing. Cambridge Universisty Press, Cambridge (1987)
8. Hovy, E.H., Marcus, M., Palmer, M., Pradhan, S., Ramshaw, L., and Weischedel, R. OntoNotes: The 90% Solution. Human Language Technology / North American Association of Computational Linguistics conference, New York, 57–60 (2006)
9. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem, Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufmann, 121–129, (1994)
10. Kohavi, R., and John, G.H. Automatic Parameter Selection by Minimizing Estimated Error. 12th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA (1995)
11. Kramer, S., Lavrac, N., and Flach, P. Propositionalization Approaches to Relational Data Mining. Relational Data Mining, S. Dzeroski and N. Lavrac (eds), Springer 262–291 (2001)

12. Lamjiri, A., Demerdash, O., Kosseim, F. Simple features for statistical Word Sense Disambiguation. SENSEVAL-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona 133–136 (2004)
13. Landwehr, N., Passerini, A., Raedt, L. D., and Frasconi, P. kFOIL: Learning Simple Relational Kernels. In Gil, Y. and Mooney, R., editors, Proc. Twenty-First National Conference on Artificial Intelligence (2006).
14. Lavrac, N., Dzeroski, S., and Grobelnik, M. Learning nonrecursive definitions of relations with LINUS. Technical report, Jozef Stefan Institute (1990)
15. Lesk, M. Automated Sense Disambiguation Using Machine-readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. SIGDOC Conference, Toronto, 24–26 (1986)
16. Lin, D. Principle based parsing without overgeneration. 31st Annual Meeting of the Association for Computational Linguistics, Columbus, 112–120 (1993)
17. McRoy, S. Using Multiple Knowledge Sources for Word Sense Discrimination. Computational Linguistics, 18(1):1–30 (1992)
18. Mihalcea, R., Chklovski, T., Kilgariff, A. The SENSEVAL-3 English Lexical Sample Task. SENSEVAL-3: 3rd International Workshop on the Evaluation of Systems for Semantic Analysis of Text, Barcelona 25–28 (2004)
19. Miller, G.A., Beckwith, R.T., Fellbaum, C.D., Gross, D., Miller, K. Wordnet: An On-line Lexical Database. International Journal of Lexicography, 3(4):235–244 (1990)
20. Mohammad, S. and Pedersen, T. Complementarity of Lexical and Simple Syntactic Features: The SyntaLex Approach to SENSEVAL-3. SENSEVAL-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, 159–162 (2004)
21. Muggleton, S. Inductive Logic Programming: derivations, successes and shortcomings. SIGART Bulletin 5(1):5–11 (1994)
22. Muggleton, S. and Raedt, L. D. Inductive logic programming: Theory and methods. Journal of Logic Programming 19,20:629–679 (1994)
23. Muggleton,S., Lodhi,H., Amini,A., Sternberg,M.J.E.: Support vector inductive logic programming, 8th International Conference on Discovery Science, Springer-Verlag, 163–175, (2005)
24. Nienhuys-Cheng, S. and de Wolf, R. Foundations of Inductive Logic Programming. Springer, Berlin (1997)
25. Paes, A., Zaverucha, G., Page, C.D. Jr., and Srinivasan, A.: ILP through propositionalization and stochastic k-term DNF learning. Sense Disambiguation using Inductive Logic Programming. Selected papers from the 16th International Conference on Inductive Logic Programming, LNCS 4455, Springer-Verlag, 379–393 (2007)
26. Parker, J.; Stahel, M. Password: English Dictionary for Speakers of Portuguese. Martins Fontes, São Paulo (1998)
27. Pedersen, T. A Baseline Methodology for Word Sense Disambiguation. 3rd International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City (2002)

28. Pradhan, S., Loper, E., Dligach, D. and Palmer, M. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. Fourth International Workshop on Semantic Evaluations, Prague, 87–92 (2007)
29. Procter, P. (editor). Longman Dictionary of Contemporary English. Longman Group, Essex (1978)
30. Quillian, M.R. A Design for an Understanding Machine. Colloquium of semantic problems in natural language. Cambridge University, Cambridge (1961)
31. Ratnaparkhi, A. A Maximum Entropy Part-Of-Speech Tagger. Empirical Methods in NLP Conference, University of Pennsylvania (1996)
32. Schutze, H. Automatic Word Sense Discrimination. Computational Linguistics, 24(1):97–124 (1998)
33. Siegel, S. Nonparametric Statistics for the Behavioural Sciences. McGraw-Hill, New York (1956)
34. Specia, L.: A Hybrid Relational Approach for WSD - First Results. Student Research Workshop at Coling-ACL, Sydney, 55–60, (2006)
35. Specia, L., Nunes, M.G.V., Stevenson, M. Learning Expressive Models for Word Sense Disambiguation. 45th Annual Meeting of the Association for Computational Linguistics, Prague, 41–48 (2007)
36. Specia, L., Nunes, M.G.V., Srinivasan, A., Ramakrishnan, G. Word Sense Disambiguation using Inductive Logic Programming. Selected papers from the 16th International Conference on Inductive Logic Programming, LNCS 4455, Springer-Verlag, 409–423 (2007)
37. Specia, L., Nunes, M.G.V., Srinivasan, A., Ramakrishnan, G. USP-IBM-1 and USP-IBM-2: The ILP-based Systems for Lexical Sample WSD in SemEval-2007. 4th International Workshop on Semantic Evaluations, Prague, 442–445 (2007)
38. Specia, L, Nunes, M.G.V., and Stevenson, M. Exploiting Parallel Texts to Produce a Multilingual Sense-tagged Corpus for Word Sense Disambiguation. RANLP-05, Borovets, 525–531 (2005)
39. Srinivasan, A. The Aleph Manual. Available at http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/ (1999)
40. Stevenson, M. and Wilks, Y. The Interaction of Knowledge Sources for Word Sense Disambiguation. Computational Linguistics, 27(3):321–349 (2001)
41. Wilks, Y. and Stevenson, M. Combining Independent Knowledge Sources for Word Sense Disambiguation. 3rd Conference on Recent Advances in Natural Language Processing, Tzigov Chark, 1–7 (1997)
42. Wilks, Y. and Stevenson, M. The Grammar of Sense: Using Part-of-speech Tags as a First Step in Semantic Disambiguation. Natural Language Engineering, 4(1):1–9 (1998)
43. Yarowsky, D. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. 33rd Annual Meeting of the Association for Computational Linguistics, Cambridge, 189–196 (1995)
44. Zelezny, F., Srinivasan, A., and Page C.D. Jr.: Randomised restarted search in ILP. Machine Learning 64(1-3): 183-208 (2006)

A Tabulation of Results

Word	Majority Class	Baseline	ILP-S	ILP-D
activate	82.46 ± 3.56	81.58 ± 3.63	82.45 ± 3.56	92.98 ± 2.39
add	45.80 ± 4.35	81.68 ± 3.38	83.21 ± 3.27	85.50 ± 3.08
appear	44.70 ± 4.33	70.45 ± 3.97	71.21 ± 3.94	88.64 ± 2.76
ask	27.78 ± 3.99	53.97 ± 4.44	53.17 ± 4.45	60.32 ± 4.36
begin	59.74 ± 5.59	63.64 ± 5.48	72.73 ± 5.08	74.03 ± 5.00
climb	55.22 ± 6.08	71.64 ± 5.51	86.57 ± 4.17	85.07 ± 4.35
decide	67.74 ± 5.94	79.03 ± 5.17	80.64 ± 5.02	79.03 ± 5.17
eat	88.37 ± 3.46	87.21 ± 3.60	88.37 ± 3.46	87.21 ± 3.60
encounter	50.77 ± 6.20	73.85 ± 5.45	72.30 ± 5.55	73.85 ± 5.45
expect	74.36 ± 4.94	79.49 ± 4.57	92.31 ± 3.02	92.31 ± 3.02
express	69.09 ± 6.23	67.27 ± 6.33	67.27 ± 6.33	78.18 ± 5.57
hear	46.88 ± 8.82	62.50 ± 8.56	62.50 ± 8.56	65.63 ± 8.40
lose	52.78 ± 8.32	52.78 ± 8.32	52.70 ± 8.32	52.78 ± 8.32
mean	52.50 ± 7.90	72.50 ± 7.06	75.00 ± 6.85	75.00 ± 6.85
miss	33.33 ± 8.61	36.67 ± 8.80	36.66 ± 8.80	40.00 ± 8.94
note	38.81 ± 5.95	53.73 ± 6.09	88.06 ± 3.96	88.06 ± 3.96
operate	16.67 ± 8.78	66.67 ± 11.11	72.22 ± 10.56	72.22 ± 10.56
play	46.15 ± 6.91	53.85 ± 6.91	55.77 ± 6.89	55.77 ± 6.89
produce	52.13 ± 5.15	63.83 ± 4.96	65.96 ± 4.89	74.47 ± 4.50
provide	85.51 ± 4.24	89.86 ± 3.63	86.96 ± 4.05	89.86 ± 3.63
receive	88.89 ± 6.05	88.89 ± 6.05	88.89 ± 6.05	88.89 ± 6.05
remain	78.57 ± 4.90	84.29 ± 4.35	85.71 ± 4.18	87.14 ± 4.00
rule	50.00 ± 9.13	66.67 ± 8.61	83.33 ± 6.80	86.67 ± 6.21
smell	40.74 ± 6.69	79.63 ± 5.48	75.92 ± 5.82	74.07 ± 5.96
suspend	35.94 ± 6.00	56.25 ± 6.20	56.25 ± 6.20	57.81 ± 6.17
talk	72.60 ± 5.22	73.97 ± 5.14	73.97 ± 5.14	73.97 ± 5.14
treat	28.07 ± 5.95	47.37 ± 6.61	57.89 ± 6.54	50.88 ± 6.62
use	71.43 ± 12.07	92.86 ± 6.88	92.86 ± 6.88	92.86 ± 6.88
wash	67.65 ± 8.02	73.53 ± 7.57	61.76 ± 8.33	64.71 ± 8.20
watch	74.51 ± 6.10	76.47 ± 5.94	72.54 ± 6.25	74.51 ± 6.10
win	44.74 ± 8.07	47.37 ± 8.10	57.89 ± 8.01	60.53 ± 7.93
write	26.09 ± 9.16	47.83 ± 10.42	39.13 ± 10.18	52.17 ± 10.42
Micro	56.26 ± 1.12	69.94 ± 1.03	73.14 ± 1.00	76.60 ± 0.95
Macro	55.31 ± 1.12	68.67 ± 1.05	71.63 ± 1.02	74.22 ± 0.99

Fig. 14. Estimates of accuracies of disambiguation models for datasets in Monolingual (1) task. "Micro" and "Macro" refer to averages weighted, and un-weighted by the number of examples.

Word	Majority Class	Baseline	ILP-S	ILP-D
allow	97.14 ± 2.82	97.14 ± 2.82	97.14 ± 2.82	100.00 ± 0.00
announce	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
approve	91.67 ± 7.98	91.67 ± 7.98	91.67 ± 7.98	91.67 ± 7.98
area	70.27 ± 7.51	62.16 ± 7.97	62.16 ± 7.97	62.16 ± 7.97
ask	51.72 ± 6.56	50.00 ± 6.57	94.83 ± 2.91	94.83 ± 2.91
authority	23.81 ± 9.29	47.62 ± 10.90	57.14 ± 10.80	90.48 ± 6.41
base	10.00 ± 6.71	30.00 ± 10.25	35.00 ± 10.67	30.00 ± 10.25
begin	56.25 ± 7.16	45.83 ± 7.19	33.33 ± 6.80	35.42 ± 6.90
believe	78.18 ± 5.57	67.27 ± 6.33	100.00 ± 0.00	100.00 ± 0.00
bill	75.49 ± 4.26	66.67 ± 4.67	64.71 ± 4.73	64.71 ± 4.73
build	73.91 ± 6.47	60.87 ± 7.20	56.52 ± 7.31	69.57 ± 6.78
buy	76.09 ± 6.29	69.57 ± 6.78	91.30 ± 4.15	91.30 ± 4.15
capital	96.49 ± 2.44	96.49 ± 2.44	96.49 ± 2.44	98.25 ± 1.74
care	28.57 ± 17.07	42.86 ± 18.70	42.86 ± 18.70	42.86 ± 18.70
carrier	71.43 ± 9.86	71.43 ± 9.86	71.43 ± 9.86	71.43 ± 9.86
chance	40.00 ± 12.65	20.00 ± 10.33	20.00 ± 10.33	20.00 ± 10.33
claim	80.00 ± 10.33	80.00 ± 10.33	80.00 ± 10.33	80.00 ± 10.33
come	23.26 ± 6.44	20.93 ± 6.20	20.93 ± 6.20	20.93 ± 6.20
complain	85.71 ± 9.35	85.71 ± 9.35	85.71 ± 9.35	85.71 ± 9.35
complete	93.75 ± 6.05	93.75 ± 6.05	93.75 ± 6.05	93.75 ± 6.05
condition	76.47 ± 7.27	73.53 ± 7.57	79.41 ± 6.93	100.00 ± 0.00
contribute	50.00 ± 11.79	38.89 ± 11.49	50.00 ± 11.79	100.00 ± 0.00
defense	28.57 ± 9.86	14.29 ± 7.64	19.05 ± 8.57	14.29 ± 7.64
describe	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
development	62.07 ± 9.01	62.07 ± 9.01	58.62 ± 9.15	58.62 ± 9.15
do	90.16 ± 3.81	90.16 ± 3.81	90.16 ± 3.81	100.00 ± 0.00
drug	86.96 ± 4.97	76.09 ± 6.29	69.57 ± 6.78	71.74 ± 6.64
effect	76.67 ± 7.72	63.33 ± 8.80	60.00 ± 8.94	60.00 ± 8.94
end	52.38 ± 10.90	47.62 ± 10.90	47.62 ± 10.90	52.38 ± 10.90
enjoy	57.14 ± 13.23	50.00 ± 13.36	57.14 ± 13.23	50.00 ± 13.36
examine	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
exchange	73.77 ± 5.63	63.93 ± 6.15	63.93 ± 6.15	63.93 ± 6.15
exist	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
explain	88.89 ± 7.41	88.89 ± 7.41	83.33 ± 8.78	83.33 ± 8.78
feel	68.63 ± 6.50	68.63 ± 6.50	100.00 ± 0.00	100.00 ± 0.00
find	82.14 ± 7.24	78.57 ± 7.75	82.14 ± 7.24	85.71 ± 6.61
future	86.30 ± 2.85	77.40 ± 3.46	85.62 ± 2.90	92.47 ± 2.18
go	45.90 ± 6.38	32.79 ± 6.01	37.70 ± 6.21	39.34 ± 6.25
grant	80.00 ± 17.89	80.00 ± 17.89	100.00 ± 0.00	100.00 ± 0.00
hold	37.50 ± 9.88	25.00 ± 8.84	54.17 ± 10.17	83.33 ± 7.61
hour	89.58 ± 4.41	85.42 ± 5.09	89.58 ± 4.41	75.00 ± 6.25
job	82.05 ± 6.15	76.92 ± 6.75	97.44 ± 2.53	97.44 ± 2.53
join	38.89 ± 11.49	38.89 ± 11.49	55.56 ± 11.71	100.00 ± 0.00
keep	56.25 ± 5.55	46.25 ± 5.57	75.00 ± 4.84	92.50 ± 2.94
kill	87.50 ± 8.27	87.50 ± 8.27	87.50 ± 8.27	87.50 ± 8.27
lead	38.46 ± 7.79	30.77 ± 7.39	43.59 ± 7.94	97.44 ± 2.53
maintain	90.00 ± 9.49	80.00 ± 12.65	80.00 ± 12.65	80.00 ± 12.65
management	71.11 ± 6.76	62.22 ± 7.23	57.78 ± 7.36	60.00 ± 7.30
move	97.87 ± 2.10	97.87 ± 2.10	97.87 ± 2.10	97.87 ± 2.10
need	71.43 ± 6.04	57.14 ± 6.61	57.14 ± 6.61	58.93 ± 6.57
network	90.91 ± 3.88	76.36 ± 5.73	78.18 ± 5.57	74.55 ± 5.87
occur	86.36 ± 7.32	77.27 ± 8.93	77.27 ± 8.93	95.45 ± 4.44
order	91.23 ± 3.75	91.23 ± 3.75	96.49 ± 2.44	96.49 ± 2.44
part	66.20 ± 5.61	50.70 ± 5.93	95.77 ± 2.39	97.18 ± 1.96
people	90.43 ± 2.74	86.09 ± 3.23	100.00 ± 0.00	100.00 ± 0.00
plant	98.44 ± 1.55	98.44 ± 1.55	98.44 ± 1.55	96.88 ± 2.17
point	81.33 ± 3.18	70.00 ± 3.74	78.00 ± 3.38	89.33 ± 2.52
policy	97.44 ± 2.53	97.44 ± 2.53	100.00 ± 0.00	100.00 ± 0.00
position	46.67 ± 7.44	28.89 ± 6.76	91.11 ± 4.24	91.11 ± 4.24
power	27.66 ± 6.52	53.19 ± 7.28	53.19 ± 7.28	55.32 ± 7.25
prepare	77.78 ± 9.80	55.56 ± 11.71	61.11 ± 11.49	66.67 ± 11.11
president	72.88 ± 3.34	68.36 ± 3.50	68.36 ± 3.50	67.80 ± 3.51
produce	75.00 ± 6.53	72.73 ± 6.71	95.45 ± 3.14	95.45 ± 3.14
promise	75.00 ± 15.31	75.00 ± 15.31	62.50 ± 17.12	75.00 ± 15.31
propose	85.71 ± 9.35	85.71 ± 9.35	85.71 ± 9.35	85.71 ± 9.35
prove	68.18 ± 9.93	45.45 ± 10.62	50.00 ± 10.66	45.45 ± 10.62
raise	14.71 ± 6.07	8.82 ± 4.86	8.82 ± 4.86	20.59 ± 6.93
rate	86.21 ± 2.86	79.31 ± 3.36	100.00 ± 0.00	100.00 ± 0.00
recall	86.67 ± 8.78	86.67 ± 8.78	86.67 ± 8.78	86.67 ± 8.78
receive	95.83 ± 2.88	95.83 ± 2.88	93.75 ± 3.49	95.83 ± 2.88
regard	71.43 ± 12.07	78.57 ± 10.97	71.43 ± 12.07	78.57 ± 10.97
remember	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
replace	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
report	91.43 ± 4.73	91.43 ± 4.73	91.43 ± 4.73	91.43 ± 4.73
rush	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
say	98.71 ± 0.49	98.52 ± 0.52	99.63 ± 0.26	99.63 ± 0.26
see	44.44 ± 6.76	37.04 ± 6.57	40.74 ± 6.69	35.19 ± 6.50
set	28.57 ± 6.97	19.05 ± 6.06	19.05 ± 6.06	21.43 ± 6.33
share	97.14 ± 0.73	95.62 ± 0.89	98.10 ± 0.60	98.10 ± 0.60
source	37.14 ± 8.17	25.71 ± 7.39	22.86 ± 7.10	25.71 ± 7.39
space	78.57 ± 10.97	92.86 ± 6.88	78.57 ± 10.97	78.57 ± 10.97
system	48.57 ± 5.97	40.00 ± 5.86	98.57 ± 1.42	98.57 ± 1.42
turn	38.71 ± 6.19	22.58 ± 5.31	27.42 ± 5.67	22.58 ± 5.31
value	98.31 ± 1.68	94.92 ± 2.86	100.00 ± 0.00	100.00 ± 0.00
work	55.81 ± 7.57	46.51 ± 7.61	48.84 ± 7.62	48.84 ± 7.62
Micro	78.10 ± 0.62	73.41 ± 0.66	80.58 ± 0.59	83.15 ± 0.56
Macro	71.69 ± 0.67	67.53 ± 0.70	73.56 ± 0.66	77.40 ± 0.62

Fig. 15. Estimates of accuracies of disambiguation models for datasets in Monolingual (2) task.

Word	Majority Class	Baseline	ILP-S	ILP-D
come	50.30 ± 7.45	60.00 ± 7.30	75.56 ± 6.41	75.56 ± 6.41
get	21.00 ± 6.52	30.77 ± 7.39	64.10 ± 7.68	76.92 ± 6.75
give	88.80 ± 4.70	95.56 ± 3.07	100.00 ± 0.00	100.00 ± 0.00
go	68.50 ± 6.64	73.47 ± 6.31	77.55 ± 5.96	75.51 ± 6.14
look	50.30 ± 7.29	78.72 ± 5.97	76.59 ± 6.18	80.85 ± 5.74
make	70.00 ± 7.07	78.57 ± 6.33	78.57 ± 6.33	90.48 ± 4.53
take	28.50 ± 7.98	43.75 ± 8.77	56.25 ± 8.77	53.13 ± 8.82
Micro	55.69 ± 2.87	67.56 ± 2.71	76.59 ± 2.45	79.93 ± 2.32
Macro	53.91 ± 2.88	65.83 ± 2.74	75.52 ± 2.49	78.92 ± 2.36

Fig. 16. Estimates of accuracies of disambiguation models for datasets in the bilingual task.