Computers and the Humanities, special issue on Senseval
34:1-2, 171-177, 2000

Memory-Based Word Sense Disambiguation

Jorn Veenstra, Antal van den Bosch, Sabine Buchholz, Walter Daelemans & Jakub Zavrel
(`[veenstra,antalb,buchholz,walter,zavrel]@kub.nl`)
ILK, Tilburg University, PO-box 90153, 5000 LE Tilburg, the Netherlands

Abstract. We describe a memory-based classification architecture for word sense disambiguation and its application to the SENSEVAL evaluation task. For each ambiguous word, a semantic word expert is automatically trained using a memory-based approach. In each expert, selecting the correct sense of a word in a new context is achieved by finding the closest match to stored examples of this task. Advantages of the approach include (i) fast development time for word experts, (ii) easy and elegant automatic integration of information sources, (iii) use of all available data for training the experts, and (iv) relatively high accuracy with minimal linguistic engineering.

1. Introduction

In this paper we describe a memory-based approach to training word experts for *word sense disambiguation* (WSD) as defined in the SENSEVAL task: the association of a word in context with its contextually appropriate sense tag. In our current system, training of the semantic word experts is based on POS-tagged corpus examples and selected information from dictionary entries. The general approach is completely automatic; it only relies on the availability of a relatively small number of annotated examples for each sense of each word to be disambiguated, and not on human linguistic or lexicographic intuitions. It is therefore easily adaptable and portable.

Memory-Based Learning (MBL) is a classification-based, supervised learning approach. In this framework, a WSD problem has to be formulated as a classification task: given a set of feature values describing the context in which the word appears and any other relevant information as input, a *classifier* has to select the appropriate output class from a finite number of a priori given classes. In our approach, we construct a distinct classifier for each word to be disambiguated. We interpret this classifier as a word-expert (Berleant, 1995). Alternative supervised learning algorithms could be used to construct such word experts. The distinguishing property of memory-based learning as a classification-based supervised learning method is that it does not abstract from the training data the way that alternative learning methods (e.g. decision tree learning, rule induction, or neural networks) do.

In the remainder of this paper, we describe the different memory-based learning algorithms used, discuss the setup of our memory-based classification architecture for WSD, and report the generalization accuracy on the SENSEVAL data both for cross-validation on the training data, and for the final run on the evaluation data.

2. Memory-Based Learning

MBL keeps all training data in memory and only abstracts at classification time by extrapolating a class from the most similar item(s) in memory (i.e. it is a *lazy* learning method instead of the more common *eager* learning approaches). In recent work (Daelemans et al., 1999) we have shown that for typical natural language processing tasks, this lazy learning approach is at an advantage because it "remembers" exceptional, low-frequency cases which are nevertheless useful to extrapolate from. Eager learning methods "forget" information, because of their pruning and frequency-based abstraction methods. Moreover, the automatic feature weighting in the similarity metric of a memory-based learner makes the approach well-suited for domains with large numbers of features from heterogeneous sources, as it embodies a smoothing-by-similarity method when data is sparse (Zavrel and Daelemans, 1997). For our experiments we have used TiMBL[1], an MBL software package developed in our group (Daelemans et al., 1998). TiMBL includes the following variants of MBL:

IB1: The distance between a test item and each memory item is defined as the number of features for which they have a different value (overlap metric).

IB1-IG: In most cases, not all features are equally relevant for solving the task; this variant uses information gain (an information-theoretic notion measuring the reduction of uncertainty about the class to be predicted when knowing the value of a feature) to weight the cost of a feature value mismatch during comparison.

IB1-MVDM: For typical symbolic (nominal) features, values are not ordered. In the previous variants, mismatches between values are all interpreted as equally important, regardless of how similar (in terms of classification behaviour) the values are. We adopted the *modified value difference metric* to assign a different distance between each pair of values of the same feature.

MVDM-IG: MVDM with IG weighting.

IGTREE: In this variant, an oblivious decision tree is created with features as tests, and ordered according to information gain of features,

[1] TiMBL is available from: http://ilk.kub.nl/.

as a heuristic approximation of the computationally more expensive pure MBL variants.

For more references and information about these algorithms we refer to (Daelemans et al., 1998; Daelemans et al., 1999).

3. System Architecture and Experiments

For the WSD task, we train classifiers for each word to be sense-tagged[2]. To settle on an optimal memory-based learning algorithm variant (i.e. IB1, IB1-IG, IB1-MVDM, or IGTREE) and parameters (e.g. k, the number of similar items taken into account when extrapolating from memory), as well as different possible feature construction settings (see below), ten-fold cross-validation is used: the training data is split into ten equal parts, and each part in turn is used as a test set, with the remaining nine parts as training set. All sensible parameter settings, algorithm variants, and feature construction settings are tested, and those settings giving the best results in the cross-validation are used to construct the final classifier, this time based on all available training data. This classifier is then tested on the SENSEVAL test cases for that word.

Feature Extraction The architecture described is suited for WSD in general, and this can include various types of distinctions ranging from rough senses that correspond to a particular POS tag, to very fine distinctions for which semantic inferences need to be drawn from the surrounding text. The 36 words and their senses in the SENSEVAL task embody many such different types of disambiguations. Since we do not know beforehand what features will be useful for each particular word and its senses, and because our classifier can automatically assess feature relevance, we have chosen to include a number of different information sources in the representation for each case. All information is taken from the dictionary entries in the HECTOR dictionary, and from the corpus files, both of which have been labeled with Part of Speech tags using MBT, our Memory-Based Tagger (Daelemans et al., 1996). We did not use any further information such as external lexicons or thesauri.

The sentences in the corpus files contain sense-tagged examples of the word in context. For example:

`800002 An image of earnest Greenery is almost tangible. Eighteen years ago she lost one of her six children in an` < *tag_532675* > *accident* < / >

[2] In some cases, the SENSEVAL task requires sense-tagging a word/POS-tag combination; we will refer to both situations as word sense-tagging.

```
on Stratford Road, a tragedy which has become a pawn in the pitiless point-scoring
of small-town vindictiveness.
```

The dictionary contains a number of fields for each sense, some of which (i.e. the ‘ex’ (example) and ‘idi’ (idiom) fields) are similar to the corpus examples. These underwent the same treatment as the corpus examples: these cases were used to extract both context features (directly neighbouring words and POS-tags, as described in section 3), and keyword features (informative words from a wide neighbourhood; see section 3). The only other field from the dictionary that we used is the ‘def’ field, which gives a definition for a sense. During the cross-validation, the examples which originated from the dictionary were always kept in the training portion of the data to have a better estimate of the generalization error. Note that for both dictionary and corpus examples, we took the sense-tag that it was labeled with as a literal atom[3], and did not take into account the hierarchical sense/sub-sense structure of the category labels. All cases that were labeled as errors or omissions (i.e. the 999997 and 999998 tags) were discarded. Disjunctions were split into (two) separate cases.

Context Features We used the word form and the Part-of-Speech (POS) tag of the word of interest and the surrounding positions as features. After some initial experiments, the size of the window was set to two words to the left and to the right. This gives the following representation for the example given above:

```
800002,in,IN,an,DT,accident,NN,on,IN,Stratford,53267
```

Keyword Features Often the direct context cannot distinguish between two senses. In such cases it is useful to look at a larger context (e.g. the whole text snippet that comes with the example) to guess the meaning from its content words. As there is a large number of possible content words, and each sentence contains a different number of them, it is not practical to represent all of them in the fixed-length feature-value vector that is required by the learning algorithm. We therefore used only a limited set of “informative” words, extracted from i) sentences in the corpus file and ii) the ‘ex’ and ‘idi’ sentences in the dictionary file, we will call these words the *keywords*. The method is essentially the same as in the work of Ng and Lee (1996), and extracts a number of keywords per sense. These keywords are then used as binary features, which take the value 1 if the word is present in the example, and the value 0 if it is not. A word is a keyword for a sense if it obeys the following three properties: (i) the word occurs in more than $M1$

[3] Although we did strip the letter suffixes (such as `-x`), except for the `-p` suffix.

percent of the cases with the sense; a high value of $M1$ thus restricts the keywords to those that are very specific for a particular sense, (ii) the word occurs at least $M2$ times in the corpus; a high value of $M2$ thus eliminates low-frequent keywords, (iii) only the $M3$ most frequently occurring keywords for a sense are extracted, restricting somewhat the number of keywords that are extracted for very frequent senses.

Definition Features In addition to the keywords that passed the above selection, we use all open class words (nouns, adjectives, adverbs and verbs) in the 'def' field of the dictionary entry as features. Comparable to the keyword feature the definition word feature has the value '1' if it occurs in the test sentence else it has the value '0'. The 'def' field is only used for this purpose, and is not converted to a training case.

After the addition of both types of keywords, a complete case for our example will look as follows:

```
800002,in,IN,an,DT,accident,NN,on,IN,Stratford,NNP,0,0,... ...,0,0,0,1,0,0,0,0,0,0,0,532675
```

Post-processing The 'dict' files contain information about multi-word expressions, compounds or collocations of a word related to a specific sense, e.g. the collocation 'golden handshake' strongly predicts sense '516773'. Using this information in a post-processing step gave a slight improvement in performance.

Results In this section we present the results we obtained with the optimal choice of metrics and feature construction parameters found with 10-fold cross validation on the training data, and the results on the evaluation data, as measured by the SENSEVAL coordination team. For comparison we also provide the baseline results (on the training data), obtained by always choosing the most frequent sense.

Table I shows the results per word. The algorithm and metric applied are indicated in the metric column; the value of k in the third column; the values of $M1$, $M2$ and $M3$ in the next column; the accuracy with the optimal settings can be found in the 'train.opt' column; and the accuracy obtained with the default setting (M1=0.8, M2=5, M3=5; the default suggested by Ng & Lee; 1996) and algorithm (IB1-MVDM, k=1, no weighting) is given in the column 'train.def'. The three rightmost columns give the scores on the evaluation data, measured by the fine-grained, medium, and coarse standard respectively. For an overview of the scoring policy and a comparison to other systems participating in SENSEVAL we refer to Rosenzweig (1999).

4. Conclusion

A memory-based architecture for word sense disambiguation does not require any hand-crafted linguistic knowledge, but only annotated training examples. Since for the present SENSEVAL task dictionary information was available, we made use of this as well, and it was easily accommodated in the learning algorithm.

We believe that MBL is well-suited to domains such as WSD, where large numbers of features and sparseness of data interact to make life difficult for many other (e.g. probabilistic) machine-learning methods, and where nonetheless even very infrequent or exceptional information may prove to be essential for good performance. However, since this work presents one of the first (but cf. Ng and Lee (1996) and Wilks and Stevenson (1998)) excursions of MBL techniques into WSD territory, this claim needs further exploration.

Although the work presented here is similar to many other supervised learning approaches, and in particular to the Exemplar-based method used by (Ng and Lee, 1996) (which is essentially IB1-MVDM with k=1), the original aspect of the work presented in this paper lies in the fact that we have used a cross-validation step per word to determine the optimal parameter-setting, yielding an estimated performance improvement of 14.4 % over their default setting.

References

Berleant, D.: 1995, ‘Engineering word-experts for word disambiguation’. *Natural Language Engineering* pp. 339–362.

Daelemans, W., A. Van den Bosch, and J. Zavrel: 1999, ‘Forgetting exceptions is harmful in language learning’. *Machine Learning, Special issue on Natural Language Learning*.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch: 1998, ‘TiMBL: Tilburg Memory Based Learner, version 1.0, Reference Guide’. *ILK Technical Report 98-03, available from: http://ilk.kub.nl/*.

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis: 1996, ‘MBT: A Memory-Based Part of Speech Tagger Generator’. In: E. Ejerhed and I. Dagan (eds.): *Proc. of Fourth Workshop on Very Large Corpora*. pp. 14–27.

Ng, H. T. and H. B. Lee: 1996, ‘Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach’. In: *Proc. of 34th meeting of the Assiociation for Computational Linguistics*.

Rosenzweig, S.: 1999, ‘SENSEVAL SCORING???’. *Computers and the Humanities* **9999**, 000–999.

Wilks, Y. and M. Stevenson: 1998, ‘Word Sense Disambiguation using Optimised Combinations of Knowledge Sources’. In: *Proceedings of COLING-ACL’98*. Montreal, Quebec, Canada, pp. 1398–1402.

Zavrel, J. and W. Daelemans: 1997, ‘Memory-Based Learning: Using Similarity for Smoothing’. In: *Proc. of 35th annual meeting of the ACL*. Madrid.

Acknowledgements

This research was done in the context of the “Induction of Linguistic Knowledge” (ILK) research programme, which is funded by the Netherlands Foundation for Scientific Research (NWO).

Table I. The best scoring metrics and parameter settings found after 10-fold cross-validation on the training set (see text). The scores are the baseline, the default and optimal settings on the training set (average of 10-fold cross-validation), and the fine-grained, medium and coarse scores on the evaluation set respectively. The scores on the evaluation set were computed by the SENSEVAL coordinators. The average scores are computed over the percentages in this table

word	metric	k	M1-M2-M3	baseline	train.def	train.opt	eval.f	eval.m	eval.c
accident	MVDM	3	0.3-3-3	67.0	81.4	90.2	92.9	95.4	98.1
amaze	IB1-IG	1	1.0-500-0	57.9	99.7	100	97.1	97.1	97.1
band	IGTREE	-	0.5-7-4	73.0	85.4	88.8	88.6	88.6	88.6
behaviour	MVDM-IG	9	0.3-5-5	95.9	94.9	96.7	96.4	96.4	96.4
bet-n	MVDM-IG	1	0.0-5-100	25.5	56.7	71.1	65.7	72.6	75.5
bet-v	IB1-IG	3	0.7-3-3	37.3	64.3	88.6	76.9	77.8	81.2
bitter	MVDM-IG	5	0.5-5-100	30.6	57.6	59.1	65.8	66.4	66.4
bother	MVDM-IG	3	0.2-5-100	45.6	72.8	83.6	85.2	87.1	87.1
brilliant	MVDM-IG	1	0.6-2-100	47.3	57.5	58.8	54.6	62.0	62.0
bury	MVDM-IG	3	0.5-5-100	32.4	35.9	46.2	50.2	51.0	51.7
calculate	IB1-IG	7	0.7-3-3	72.0	79.2	83.2	90.4	90.8	90.8
consume	IGTREE	-	0.7-5-5	37.5	32.9	58.8	37.3	43.8	49.7
derive	MVDM	5	0.0-2-100	42.9	63.9	67.3	65.0	66.1	66.8
excess	MVDM-IG	5	0.5-1-1	29.1	82.6	89.3	84.4	86.3	88.2
float-a	IGTREE	-	0.3-3-3	61.9	57.0	73.5	57.4	57.4	57.4
float-n	MVDM-IG	1	0.8-5-5	41.3	50.8	70.2	64.0	65.3	68.0
float-v	IGTREE	-	0.4-2-100	21.0	34.2	44.0	35.4	40.6	44.1
generous	MVDM	15	0.6-5-100	32.5	44.8	49.3	51.5	51.5	51.5
giant-a	IGTREE	-	1.0-500-0	93.1	92.8	94.1	97.9	99.5	100
giant-n	MVDM-IG	5	0.2-5-100	49.4	77.2	82.6	78.8	85.6	97.5
invade	MB1-IG	3	0.1-10-1	37.5	48.0	62.7	52.7	59.2	62.3
knee	MVDM-IG	5	0.0-5-100	42.8	70.3	81.4	79.3	81.8	84.1
modest	MVDM-IG	9	0.0-5-100	58.8	61.1	67.1	70.7	72.8	75.2
onion	IB1	1	0.8-5-5	92.3	90.0	96.7	80.4	80.4	80.4
promise-n	MVDM-IG	5	0.2-5-100	59.2	63.6	75.3	77.0	83.2	91.2
promise-v	IB1-IG	3	0.5-5-10	67.4	85.6	89.8	86.2	87.1	87.9
sack-n	MVDM-IG	1	0.3-3-3	44.3	75.0	90.8	84.1	84.1	84.1
sack-v	IB1	9	1.0-500-0	98.9	97.8	98.9	97.8	97.8	97.8
sanction	MVDM-IG	1	0.5-3-3	55.2	74.9	87.4	86.3	86.3	86.3
scrap-n	IB1	1	0.4-5-100	37.0	58.3	68.3	68.6	83.3	86.5
scrap-v	IGTREE	-	0.7-3-3	90.0	88.3	91.7	85.5	97.8	97.8
seize	IGTREE	-	0.5-5-100	27.0	57.1	68.0	59.1	59.1	63.7
shake	MVDM-IG	7	0.2-5-100	24.7	71.5	73.3	68.0	68.5	69.4
shirt	IGTREE	-	0.7-5-5	56.9	83.7	91.2	84.4	91.8	96.7
slight	IB1-IG	1	0.3-3-3	66.8	92.7	93.0	93.1	93.3	93.6
wooden	IGTREE	-	0.5-1-1	95.3	97.3	98.4	94.4	94.9	94.9
average				54.1	70.5	78.6	75.1	77.9	79.7