Case-Sensitivity of Classifiers for WSD: Complex Systems Disambiguate Tough Words Better

Harri M.T. Saarikoski[1], Steve Legrand[2], and Alexander Gelbukh[3]

[1] KIT Language Technology Doctorate School, Helsinki University, Finland
Harri.Saarikoski@helsinki.fi
[2] Department of Computer Science, University of Jyväskylä, Finland
stelegra@cc.jyu.fi
[3] Instituto Politecnico Nacional, Mexico City, Mexico
gelbukh@gelbukh.com

Abstract. We present a novel method for improving disambiguation accuracy by building an optimal ensemble (OE) of systems where we predict the best available system for target word using a priori case factors (e.g. amount of training per sense). We report promising results of a series of best-system prediction tests (best prediction accuracy is 0.92) and show that complex/simple systems disambiguate tough/easy words better. The method provides the following benefits: (1) higher disambiguation accuracy for virtually any base systems (current best OE yields close to 2% accuracy gain over Senseval-3 state of the art) and (2) economical way of building more effective ensembles of all types (e.g. optimal, weighted voting and cross-validation based). The method is also highly scalable in that it utilizes readily available factors available for any ambiguous word in any language for estimating word difficulty and defines classifier complexity using known properties only.

1 Introduction

Innumerable methods of word sense disambiguation have been tested to solve the WSD task adequately for NLP applications but no single method (e.g. classifier algorithm, configuration or ensemble, feature set or selection scheme) has been found to work superiorly for all target words. Partly because of this bias, disambiguation methods have reached a standstill [4,13]. The conclusion from this is that different disambiguation methods result in different performance results ('system bias'). In fact, differences of up to 30% in precision at word (average 5% and 3% in Senseval-2 and Senseval-3 evaluations respectively [4,13]) can take place even between state-of-the-art systems. The other way of defining this incompatibility between target words and classifying systems is that each word poses a unique set of learning problems ('word bias'). Among others [22,5,14] have showed that different classifiers for WSD have discrete but intact strong regions with regard to factors defining the word for disambiguation and the word's training contexts (e.g. number of sense-classes or number of training examples). In [6] classifier configuration through their parameters was also shown to have considerable effect on performance.

A. Gelbukh (Ed.): CICLing 2007, LNCS 4394, pp. 253 – 266, 2007.

Optimal ensembling (OE) [17] is a method for mutually solving these two biases by attempting to redirect a learning problem (target word) to the classifier most equipped to disambiguate it. The method has been outlined in general terms in [1,3] and the such a method was suggested WSD task in [14]. More specifically, OE attempts to discover *n* base systems whose accuracies are relatively strong and whose classification profile in terms of performance is as different as possible. We call such base systems *maximally complementary*. In order to select the best base system, a machine learner is then trained with prediction factors calculated from that word and its training contexts (e.g. number of senses in the word, number of features in word training data). High prediction accuracies and resulting gains over base systems (and state of the art) using this method are reported in this paper.

Optimal ensembles have largely been neglected in WSD. This is probably because it has been believed that a single system might prove to be superior or, alternatively, because of the lack of research into factors that might predict the best system out of the ones available. In fact, large majority of WSD systems implemented so far have either been single-classifier systems (e.g. SMU system based on instance-based classifier [12]) or multi-system equal voting ensembles (e.g. JHU system a voting ensemble of six classifiers trained with the same three-feature-set input [21]). In contrast with OE, these ensembles either apply the same system for all test words (using equal or weighted voting ensemble, VE) or select best base system based on cross-validation results (CV). OE in contrast uses a handful of *a priori* factors (e.g. number of positive training examples) to select the best system for each word.

To carry out OE experiments, we have developed a meta-classifier to learn the best system for target word using two machine-learning toolkits (YALE [11], Weka [20]) and Self-Organizing Map algorithm [9] (found useful in earlier WSD experiments [8]). The aim is to learn from available WSD system scores [4,13] the rules mapping system factors (e.g. classifier algorithm, feature sets) to word and training input factors (e.g. amount of training, word grain). The output from this meta-classifier is the optimal available system for that target word which WSD systems can utilize in their building multi-system ensembles (whether they be OE, VE, CV or any other).

In this paper, we define the strength of a handful of machine learners in terms of word and training input factors. Particularly we focus on Support Vector Machines and Naive Bayes learners and investigate different configurations of those classifiers (e.g. variations of SVM complexity parameter and kernel type, NB variants with system confidence meta-knowledge and binarization of sense-classes). We present a dozen best-system prediction experiments using Senseval WSD systems and our own systems.

In the next sections, we define the case factors and system factors, outline the method in detail and present the base systems for prediction tests. Results from prediction tests follow then. Final sections are dedicated to discussion, conclusions and further research required.

2 Prediction Factors

In this section, we define the case factors that predict the best system for word and define the properties of some classifiers that help define their respective strengths.

2.1 Case Factors

Performance of best system in Senseval evaluations [4,13] degrades steadily (though not linearly) when, for example, training is decreased and word grain increased [22]. Furthermore, [22] showed that strength of different classifiers also varies with amount of training, word grain and dominant sense ratio. They showed that one system tends to be strong at low values of a case factor (i.e. at tough words) and another at high values (i.e. at easy words). For instance, a simple Transformation-Based Learner (TBL) performed better when dominant sense frequency was high (70..) and NB when it was low (0..70). Some case factors split classifiers into three regions so that one is strong at low and high values and the other at middle range. For instance, Decision List outperformed two more complex Bayesian classifiers at very low (< 5) and high (< 17) values with Bayesians outperforming it in the middle range [22]. In this paper, the system-differentiating capacity of the following factors is investigated (see also [5,6,22,14,4,13]):

Amount of training (total*) aka volume. *Totalex/totalf* is the total number of feature extracted from training data depending on the selection of feature set (e.g. some hundreds from global 1-grams, some thousands from global sequential 2-grams).

Positive/negative training (pos*,neg*). *Posex/posf* is the average number of matching training examples/features per sense *s*. *Negex/negf* is the average number of non-matching training examples / features per each non-*s.* The relationship of *total** and *pos*/neg** factors is that *totalf/totalex* set consists of subsets *posf/posex* and *negf/negex* and *domall* specifies the sense-class distribution of both training examples and features. Note also that word grain used can be counted from other factors: `negex / posex + 1` or `negf / posf + 1` and can therefore be omitted.

Training distribution (domall). *Domall* accounts for distribution of training to senses s_n: $s_1^2 + s_2^2 \ldots + s_n^2$.

Instance occurrences per feature (instperfeat) aka relevance. *Instperfeat* is an approximate measure of the average relevance of extracted features. An optimal feature for recognizing a sense can be defined as one occurring frequently, pointing uniquely to one sense (e.g. collocate 'steel' pointing to the crow bar sense of *bar.n*) is more facilitating than a feature pointing to several classes (e.g. word 'and' pointing to almost any sense of any word). Most features, however, are not of such high quality, occurring either too few times to occur in test instances or too many times to point to the correct sense.

Using these case factors, we can define tough words as having low-total*, low-pos*, high-neg*, low-domall and low/high instperfeat (e.g. *call.v, carry.v*) and simple words vice versa with mid-instperfeat (e.g. *oblique.n, hearth.n*) (examples from [4]).

To show the system-differentiating of case factors, we show strengths of systems based on two classifiers (SVM and NB) in 'word space' of two case factors:

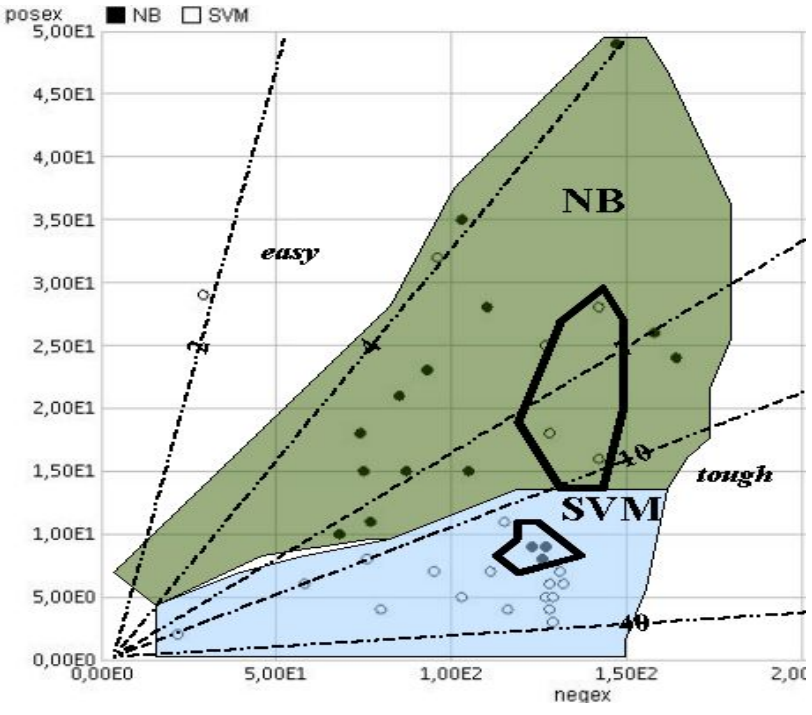

Fig. 1. Strong regions of Senseval-2 SVM/NB based systems in 'word space' of negex-posex (x,y) factors. (Plotter is provided by YALE [11]). Dots mark the words (filled = NB, unfilled = SVM). Words that lie outside linearly separable regions (i.e. most probable misclassifications) are circled. Easy words are on top left (note how at one 2-grain word SVM and NB tied) and toughness increases towards bottom right. (Posex value of *2.00E1* means average of 20 positive examples per each word sense.)

Figure 1 indicates that SVM and NB based systems occupy very different but quite linearly separable regions in negex-posex word space. We can see that SVM seems to perform well at low-posex ($3 < posex < 14$) and NB at high-posex half ($posex > 15$) while negex seems not to have as much influence. We found this relative positioning of SVM and NB to be approximately the same across different datasets and prediction tasks and also agrees with a similar experiment with the same classifiers in [5]. Interestingly, many classifier pairs tend to place in approximately those positions in word space that SVM and NB took (e.g. simple Ibk-based SMU [12] vs the more complex voting ensemble JHU [21], and TBL vs NB in [22], respectively). Some classifiers seem to handle easier words better while others tougher words.

Next we outline possible ways to define classifiers in terms of their 'complexity'.

2.2 System Factors

We exemplify some of the key differences between a handful of classifiers in terms of their respective sense decision procedures (including feature selection, weighing, evaluation and optimization strategies):

Table 1. Matrix for *adhoc* evaluation of classifier complexity. (Ibk = Instance-Based aka Nearest Neighbor Classifier, DStump = Decision Stump). (+) means that the classifier has that property (e.g. *NB-Bin+* would be NB with Bin+ property).

classifier property	SVM	NB	DTree	DStump	Ibk
Weight: performs some kind of custom weighing of features?	+	+	+	+	-
Aggreg: aggregates more than one feature (not all) per test instance to decide sense?	+	+	+/- (some)	-	+
Prior: utilizes prior sense distribution probabilities to weigh features?	-	+	-	-	-
Eval: utilizes meta-classifier's estimation of base classifier's correctness?	-	-	-	-	-
Opti: optimizes feature weights by cross-validating against training data?	+	-	-	-	-
Bin: splits n-class sense classification task into (n-1)! each vs each binary tasks?	+	-	-	-	-

This classifier matrix helps select the maximally complementary classifiers to be used as base systems in OE. We aim to prove that those differences explain why classifiers perform differently in different learning cases. The fewer positive markings (+) a classifier has, the more complex (or advanced or reflective in its decision procedure) the classifier is supposed to be. SVM is therefore proposed to be the most complex classifier of the ones compared and Ibk the simplest one with its direct comparison technique for selecting the sense of the training instance that is most similar to the test instance. Senseval system implementations of the former are UMCP [10] in Senseval-2 and IRST-kernel [13] in Senseval-3. The latter has been the base classifier with SMU [4,12] in Senseval-2 and GAMBL [13] in Senseval-3.

Specifically we focus in this paper on the differences between SVM and NB classifiers, since those have (previously been proven strong at WSD [4,13]. System matrix specifies that SVM is more 'complex' than NB in its maximization of the feature space margin by optimizing feature weights through iteration on training errors (*Opti+*), while NB makes a one-time sweep (*Opti-*) over training data and weighs features based on distribution of senses *(Prior+*) in training data. Complexity order of configurations of these and other classifiers is more difficult to estimate: for instance, whether SVM's polynomial kernel is more 'complex' than RBF kernel without knowing their performance (see [19]). In section 4, we will be looking whether the *known* system differences materialize in system performance with regard to easiness/toughness of target word and whether we can learn the complexity order of *unknown* system properties from their performance.

But first we will present the OE method in a generic WSD algorithm.

3 Optimal Ensembling Method

In this section, we integrate the OE method in a WSD algorithm (note that this method is also the setting of our prediction tests reported in the next section):

1. *Word sampling to training/testing sets.* Divide the word sample for which you have sense-tagged data (e.g. Senseval datasets [4,13]) randomly into training words (two thirds) and test words (remaining one third)[1]. Calculate case factors for all words making particularly sure that the word sample is spread equally over word space.
2. *WSD base system selection.* Select *n* strong candidate base systems that can be estimated as mutually complementary or opposite to each other (as defined in Table 1). Run cross-validation (CV) tests on training data and compare candidate system performance with regard to case factors. Then use these formal criteria for selection of two (or three) base systems to ensemble: (1) to ensure base systems are strong enough, select the base systems that have a relatively good overall CV accuracy at all words, (2) to ensure base systems are maximally complementary, select the base systems that provide the biggest gross gain (see Stage 4 for gross gain calculation) and (3) to ensure optimal prediction accuracy, check that the base system strong regions are relatively intact using word space visualizations (cf. Figure 1).
3. *Training and testing the predictor.* In order to make system regions more separable, remove the ties (i.e. training words where more than one selected base systems were within 1% of each other). This improves prediction accuracy considerably. Using the same training run data, use cross-validation tests or trial-and-error technique to discover the best machine learner for best-system prediction (predictor selection)[2]. Then run selected base systems on test words. (Note that in order to evaluate OE you do not necessarily have to develop or run the OE).
4. *Evaluation of predictor and OE.* Evaluate the ensemble performance from *PredictionAccuracy*GrossGain* where *PredictionAccuracy* is correct best-system-for-word predictions divided by total number of test words. *GrossGain* is the maximum gain of OE over best of its base systems resulting from a perfect system-for-word prediction for all test words. It is obtained by subtracting the all-words accuracy of OE from the all-words accuracy of its best base systems. For example, in a test set of two words, if system#1 wins over system#2 by 2% at word#1 and system#2 wins over system#1 by 4% at word#2, then gross gain is (2+4) / 2 = 3% *Net gain* is then calculated as follows: in a two-system OE with 0.80 prediction accuracy and 4.0% potential gain (gross gain), net gain is 0.80 * 4.0% = 3.2%. (We will also calculate gross gain *per word* as an alternate measure, see Discussion for justification of two measures).
5. *OE optimization.* Performance of OEs can be improved in various ways: (1) if prediction accuracy is low, either add training words or remove more tied words (e.g. words that have < 2% or < 3% difference between base systems) while still leaving enough words for learning to occur) or (2) if gross gain is low, reconfigure one of the base systems so that it complements the remaining base system(s) better.

4 Evaluation

In this section we report the best-system prediction experiments we carried out (including descriptions of the base systems, test settings and test results).

[1] This is the usual division for training and testing WSD systems in Senseval [4,13].

[2] We found SVM and Logistic Regression learners in their default Weka configurations [20] to generally produce the best prediction accuracies.

4.1 Test Setting

Using the method described in the previous section, we evaluated OE with systems in Senseval English lexical sample evaluations [4,13] and our own systems (separately of course). For prediction tasks with Senseval systems, we clustered systems based on the same classifier (SVM and NB). We trained best-system predictors on approximately 30-50 words (depending on the dataset and the number of removed ties, see Method) and tested the model on approximately remaining words in the dataset (typically 20-30 words).

We evaluated OEs constructed of the following types of base systems (Table 2):

Table 2. Base systems used for different optimal ensembling tasks

Dataset	Classifier	Systems [4,13]
Senseval-2	(N/A[3])	JHU [21], SMU [12], KUN [18]
	SVM	UMCP [10]
	NB	Duluth1, Duluth4 [15]
Senseval-3	SVM	IRST-kernel, nusels, TALP [13]
	NB	htsa3, CLaC1, Prob1 [13]
Custom (Senseval-2)	(N/A)	SVM [19]: c=0.1, c=1.0, c=3.0, RBF kernel NB [11], DTree [16], DStump [16,20], Ibk [1] [4]

As for our custom systems, we used SVM with polynomial kernel [19,20] varying the complexity (*c*) parameter from (values 0.1, 1.0 and 3.0). SVM's c parameter is crucial since it refers to the trade-off between training error and complexity of the training model [19] so that low *c* values prompt the SVM algorithm to build a more complex model, i.e. to iterate more over training errors. We also experimented with an alternative SVM kernel type - RBF (Gaussian based radial basis function). Though not mentioned in Table 2, we also ran a number of other types of classifiers over Senseval-2 dataset, especially two variants of NB (*NB-Bin+* and *NB-Eval+*) to compare SVM and NB strengths in their various configurations and classifier properties.

To investigate training input effects, we ran our custom systems with two contrasting types of feature sets (both from a global instance-wide window): 1-grams (i.e. collocates) that are low-volume (average at a few hundred) but high-relevance (ten instances per feature) while sequential 2-grams are high-volume (a few thousand) but low-relevance (three). These are also the two strongest feature sets for WSD [12].

4.2 Base System Complexity vs Tough / Easy Words

In this section, we will look at the ranking order of systems to confirm the effect of complexity (system properties) on different types of words (case factors).

[3] For reasons of focus, we are not describing the Senseval-2 systems in detail. Refer to quoted papers for details.

[4] We used default Weka [20] configurations for each of these classifiers. J48 implementation of C4.5 decision tree was used to evaluate DTree.

Table 3. Rank-based ordering of custom systems with regard to case factors and feature set. Order or classifiers has been established for each factor by comparing their performance at easy vs tough words (e.g. a classifier that is highest in totalf ranking did better than the others at tough words as defined by totalf factor). Other case factors were omitted either for showing similar ranking order (e.g. posex and posf compared to totalf) or contributing little to system differentation (e.g .neg*).

feature set	totalf	domall	instperfeat
1-grams	- c=0.1 - NB & Dtree & Ibk - c=1.0 - Dstump - RBF	- c=0.1 & c=1.0 & Dtree - NB & Ibk - Dstump - RBF	- Dtree - NB - Ibk - c=0.1 & c=1.0 & Dstump - RBF
2-grams	- c=0.1 - Ibk - Dtree & Dstump - NB - RBF	- c=0.1 - Dtree & Dstump & Ibk - NB - RBF	- Dtree - RBF - Dstump - NB & c=0.1 & c=1.0 - Ibk

Table 3 shows that strong correlation of 'complex' systems with tough words and 'simple' systems with easy words exists. For instance, SVM tends to be able to disambiguate tough words better, especially at low complexity parameter values (c=0.1). Most complex c value c=0.1 tends to perform well with tough words. Though not shown in the table, decadence of performance when increasing SVM *c* values (from c=0.1 toward c=3.0) is quite linear. This implies that the complexity continuum of SVM variants starts from the Gaussian RBF ('simple' system) and continues with polynomial kernel variations following the increase in values of c (complex system). Evaluating the variants of NB (NB aka NB-Prior+, NB-Bin+ and NB-Eval+), they also seem to keep their complexity order across all case factors: complex variants (NB-Bin+ and NB-Eval+) work better for tough words (almost equally well as c=0.1) and basic NB better for easier words. Such an ordering around easiness/toughness also takes place for the 'minimal pair' (DTree and DStump), differing only in the presence/absence of *Aggreg* property (i.e. DTree forms an entire decision tree, while DStump reduces that tree to a single decision node per test instance). We found the strength of Decision Stump based systems (e.g. DuluthB [15] system in Senseval-2 and in our custom batch) to be restricted to the 'easy' region of word space (top left corner in Figure 1) in contrast with its more complex ancestor Decision Tree [16] (e.g. Duluth2 [15]) which was better with tougher words (towards the center of Figure 1). Hence, it seems DTree not only seems but is also *functionally* a more complex classifier than DStump and thus suitable for tougher words. These results largely confirm the correlation of toughness and complexity.

Notably in Table 3, we find *instperfeat* (relevance) sorting the classifiers differently than the other case factors. SVM classifiers are strong in the middle range (where discriminative quality of features is highest) but DTree has risen on top (where the average instance occurrences of features is very low). We also found that while in our custom tests with little training, DTree ended 5-6% behind the best classifier, in Senseval-2 [4] DTree based systems were at a comparable level with SVM and NB

based systems. In our mind, this can only be because the more volumous and relevant training input was adequate enough for DTree to be strong at tougher words and helped it to a comparable disambiguation accuracy. NB in turn was found very ineffective (no wordwins at all in custom systems batch) to deal with 2-grams (low-relevance, high-volume). It appears that NB works better, relatively speaking, when there are enough high-occurrence features to weight with prior distributions. RBF still keeps its place towards the 'easy' end even at *instperfeat* which confirms its role as an 'easy word expert'. As with DTree, 2-grams seem to agree with RBF better than 1-grams. At 2-grams, SVM's c=0.1 configuration in fact lost considerably to RBF (while both performed equally well at 1-grams).

4.3 Base System Complexity vs Best Optimal Ensembles

In this section, we will further be looking at whether the best net gains (Table 4) are obtained from the ensembling of maximally 'opposite' simple and complex systems.

Table 4. Results from applying the method on selected base systems. For two-system predictions, best prediction accuracies, gross gains and net gains (both measures) per each system batch have been bolded. (CV = selection of best system by cross-validation, VE = equal voting ensemble). All the base systems for custom OEs (3) were trained with 1-grams only. Except in (3a2) and (3a3), predictors were trained with approximately 40 words.

System batch	Optimal ensembling task (gross gain per word / per all words)	Prediction accuracy (most freq class)	Net gain per word / per all words
(1) Senseval-2	(a) JHU+SMU (8.0 / 3.8)	0.80 (0.53)	6.4 / 3.0
	(b) SMU+KUN (8.4 / 4.1)	0.82 (0.65)	6.9 / 3.4
	(c) JHU+KUN (5.5 / 2.3)	0.75 (0.71)	4.1 / 1.7
	(d) JHU+SMU+KUN (9.5 / 5.3)	0.55 (0.39)	5.2 / 2.9
	(e) SVM(N/A)+NB (5.3 / 1.6)	0.92 (0.61)	4.9 / 1.5
(2) Senseval-3	(a) htsa3+IRSTk (4.2 / 2.0)	0.82 (0.51)	3.4 / 1.7
	(b) htsa3+nusels (3.8 / 1.6)	0.70 (0.62)	2.7 / 1.2
	(c) nusels+ IRSTk (4.5 / 1.8)	0.80 (0.60)	3.6 / 1.5
	(d) htsa3+IRSTk+nusels (6.3 / 2.3)	0.55 (0.42)	3.5 / 1.3
	(e) SVM(N/A)+NB (3.2 / 0.7)	0.90 (0.68)	2.9 / 0.6
(3) Custom	(a) SVM(c=1.0)+NB (3.9 / 1.0)	-	
	(a1) single set (40 words)	0.80 (0.63)	3.1 / *0.8*
	(a2) doubled training (70 words)	0.83 (0.63)	3.2 / 0.9
	(a3) tripled training (90 words)	0.86 (0.63)	3.4 / 1.0
	(a4) VE SVM(1.0)+NB	-	1.6 / *0.5*
	(a5) CV SVM(1.0)+NB	-	3.4 / *-1.6*
	(b) SVM(c=0.1)+NB (6.0 / 1.3)	0.82 (0.53)	4.8 / 1.1
	(c) SVM(RBF)+NB (6.1 / 1.9)	0.84 (0.57)	5.0 / 1.2
	(d) SVM(c=0.1)+SVM(RBF) (5.9 / 1.5)	0.88 (0.53)	5.2 / 1.3

Results seem to confirm the hypotheses set at the beginning of this paper. OE built from maximally opposite base systems (task 3d using the complex c=0.1 variant of SVM with the simple RBF variant) has the biggest net gain over the other OEs built out of custom systems (3a-3c). We attribute this to the capacity of the complex variant for disambiguating very tough words that also provide highest net gain per word.

Notable is also that the SVM+NB ensembles (1e, 2e) tend to get the higher prediction accuracy but lower net gains than ensembles built from individual Senseval base systems (1a-d, 2a-d) in the same datasets (1 and 2). This is probably largely due to the idiosyncracies in individual systems (with regard to training input and classifier configuration) that are by definition harder to define than base systems with known classifiers, controlled input and configuration (as in Custom systems). On the other hand, the resulting gains are low with SVM+NB ensembles because the input provided to them was probably inadequate to discriminate systems in comparison with the more 'fully trained' Senseval systems with more system variability. Also it appears that out of the classifiers evaluated for OE, SVM and NB seem to be the most complementary ones, which naturally adds to predictability between them.

High gross gains tend to correlate with high prediction accuracies and result in higher net gains. This is motivated by the fact that the regions of maximally complementary systems (i.e. ones with maximal gross gain) are easier to define by case factors and thereby ultimately predict their respective disambiguation strengths. Notice also how the more challenging three-system prediction tasks (tasks 1d and 2d) produce comparable net gains to two-system tasks (1a-1c and 2a-2c) which gives hope to reliably being able to ensemble more than two base systems. Training the predictor with more words (tasks 3a2 and 3a3) did improve prediction accuracy over single dataset training (3a1) but not very drastically. In fact, the additionally trained OE was outperformed by another system pair (c=0.1+RBF) with no extra training at all. This is evidence that even a single (Senseval or custom-generated) evaluation set is enough to predict the difference of any two *maximally complementary* base systems (almost) adequately. It should also be observed that prediction accuracies in all tasks exceed random selection baseline (0.50 in two-system tasks, 0.33 in three-system tasks) and the more challenging most frequent class baseline (*most freq class* column) which selects the most frequently winning system in training data for all test words.

5 Discussion

OE seems to produce more accurate ensembles than VE and CV ensembles when constructed of the same systems. We find conducive evidence with Senseval systems as well: for instance, DuluthC system [15] (VE of seven different systems with equal voting) ended with -2% poorer performance than its best base system. The reason for this negative net gain is probably that equal voting tends to either dilute the decisions of *very different* base systems towards average (thus ignoring that one or more of them might be more reliable in a given case) or strengthen the decision of *very similar* base systems toward their consensus decision (thus ignoring that similar classifiers let alone one single classifier cannot be right at all instances). In our experiment (3a4), VE worked almost as well as its corresponding OE (3a1) probably because SVM variant (c=1.0) and NB are neither too different (cf. close placements in Ranking table) nor too different (cf. system properties).

As to cross-validation based ensembles (CV), [1] notes that they often predict an inferior system. In our custom experiments, CV either picked low-margin winners or did not pick winners (cannot tell). This is most probably because cross-validations are run on a subset of training data, which means that the case factor values which would determine the best system (e.g. posf, negf) are then very different. Our experiment

confirmed this to be true: CV task (3a5) actually resulted in negative net gain (i.e. not even exceeding base system accuracy). Also, we should note a similar result from a hybrid of CV and VE that participated in Senseval-2: JHU system [21] (using cross-validation based weighted voting of six base classifiers), finishing -1% behind its best base classifier (still finishing first in that evaluation). With such limitations, it can be very exceptional that VE or CV (or even their hybrid) would outperform their base systems. However, results indicate that OE method's base system selection (Stage 2) based on system property matrix could be utilized to select base systems (neither too different nor too similar) for VE.

We also want to specify the justification for two net gain measures (*per all words* and *per word*). The Senseval measure [4,13] for assessing WSD system accuracy uses all-words accuracy calculated from the average of per-word accuracies (or alternatively ratio of correct/incorrect guesses at all words). This basically upweighs high-accuracy (easy) words (> 90%, e.g. *hearth.n*), which do not provide the least differences between systems in OE, since easy words are usually tied by base systems. For this reason, OE net gain per *all words* is a measure (though showing superior gains over other types ensembles) that does not do justice to OE method. This is probably the reason why the two measures do not agree on the winning OE in the dataset. Calculated using the per-word measure, OE in fact provides substantial per-word net gains. For example, if we look at the accuracy for word *drive.v* in Senseval-2 by optimal ensemble (1b), it is improved by +3% from 52% to 55% over its best base system while those base systems did not differ at an easy word *dyke.n* (90% by both systems). All-words method of counting system accuracy suggests that an improvement of that +3% at some tough word (41% -> 45%) is *less valuable* than a gain of +2% at an easy word (88% -> 90%) and since datasets have a lot of easy words, even substantial improvements at tough words are diluted in all-words accuracy. For these reasons we consider net gain *per word* as a more revealing (and reliable) measure for assessing the quality of OEs.

Even per-word net gains may be unfairly low as they are quoted in results table. This is because the words where predictor predicts the best system right are more likely to be the words where one of the base systems has had the *biggest* winning margin (> 5%) and, on the other hand, the predictor is more likely to fail at words with negligible margin (< 2%). These misclassifications are also more likely to be located on the border of base system regions (see Figure 1 outliers). This 'weighting' has not been included in the net gains but for a few prediction tasks, we confirmed this *real* net gain by looking at the actual test words correctly/incorrectly predicted. Indeed, we did find that average difference between base systems in the mispredicted words was around 3% while the average difference in correct predictions was 6%. Furthermore, the higher the prediction accuracy in the task the more likely the predictor is to predict the best system of high-margin words correctly. Until we look into it, we cannot be sure *exactly* how much further net gain these corrections will bring. Nevertheless, we estimate up to +1-4% more net gain depending on the prediction task.

6 Conclusions and Future Work

We have elaborated on a method (presented formally for WSD first in [17]) for predicting the strong regions of any ensemble of WSD systems using case factors readily available for all ambiguous words in any language. WSD system accuracies

seem to improve in all prediction tasks evaluated, which gives confidence that it works for any selection of base systems (for some naturally better). The economical benefit is that the method significantly reduces the need to perform computationally intensive and manually analyzed cross-validation tests for each target word separately.

Our main finding is that the simplicity/complexity of classifier's decision procedure (as defined in Table 1) correlates *very strongly* with easiness/toughness of the target word (as defined by case factors). We found that ensembling of 'opposite' systems (in terms of complexity) seems to produce the most successful optimal ensembles (as measured by both both prediction accuracy and net gain). We also found that some classifiers (Decision Trees and RBF kernel) are more sensitive to the *relevance* of features while others (complex SVM and NB variants) are more sensitive to the *volume* of feature pool. Apparently learning process in classifiers differs in terms of lower vs higher learning threshold (either based on volume or relevance), i.e. the point after which the classifier is able to conduct effective generalizations (decision rules) from the training data. Therefore, OE design requires not only the discovery of maximally complementary classifiers for the words but also the optimal matching of suitable training input to those classifiers. Therefore, we suggest that a fully-fledged OE-based WSD algorithm should start from finding the best feature set (using cross-validation based selection method specified in [12]) and then matching it with the best classifier (using OE method presented above).

The method can also be used 'in reverse', i.e. to estimate similarity of systems from their respective performance at easy vs tough words and thus providing a shortcut to discovering maximally complementary base systems for a 'definitive OE' (Stage 2). From Table 4 we see that JHU+KUN (task 1c) and htsa3+nusels (2b) systems, for example, are too similar to be ensembled optimally (shown in low prediction accuracies around 0.70). Case factors can also be used to estimate similarity of systems: e.g. Maximum Entropy or ME is another powerful classifier for WSD [13] but we found it largely overlapping with the strong region of NB in virtually all the case factors. Instead we should ensemble systems from extreme ends of the rank-based table and/or with opposite classifier properties. For this reason some classifier combinations (e.g. c=1.0+RBF or DStump+RBF) were not even considered.

Development of even more accurate (> 0.95) best-system predictors depends especially on correct weighing of the case and system factors (currently the factors are equally weighed). For instance, looking at the higher placements of DTree over NB in the rankings table, we can say that DTree's feature selection property (*Aggreg+*) outweighs NB's prior distribution property (*Prior+*). Also by running the evaluated classifiers on other more feature sets that are particularly relevant for WSD (e.g. PoS or syntactic features), we can confidently expect much more net gain without compromising prediction accuracy. Furthermore, training the predictor with more training words was also shown to increase prediction accuracy (3a1-3a3). Also, by accounting for remaining case factors (e.g. feature quality according to the number of senses it points, see also [1,3,14,17]), we believe it is possible to achieve almost perfect prediction accuracy for maximally complementary systems and to replicate comparable accuracy obtained with two-class predictions (0.90+) for three-class predictions (e.g. SVM+NB+DStump). We should note, however, that at least with Senseval systems [4,13], gross gain does not much improve after three systems.

With such strong correlations existing, more properties should be evaluated to define system complexity. For instance, there seems to be a difference between single-classifier

systems (e.g. SMU [12] and KUN [18]) and multi-classifier ensembles (such as JHU [21] and CS [10]) in terms of easy vs tough words. SMU and JHU systems differed in another respect as well: cross-validation based best feature set selection routine was applied in SMU while JHU employed three static feature sets (JHU). In [5] the same type of correlation with different feature selection schemes was found in a text categorization classifier experiment: simple Information Gain (based on overlap of features between training model and test instance) performed better with easier classification cases and the more complex Binormal Separation scheme (see [5] for details) was better with tougher cases. We also want to note the positive effect of extra lexical knowledge (WordNet::Domains available at http://wndomains.itc.it/wordnetdomains.html) and unsupervised clustering of extra training contexts (British National Corpus) as utilized by top Senseval-3 system (IRSTkernel) at Senseval-3 [13]. In light of current findings, we believe simpler classifiers gradually gain ground with increased training.

For most words in most languages, the cost of acquiring enough sense-tagged examples to disambiguate all words in all targeted languages to maximum accuracy (i.e. to the point when machine learning process becomes saturated) is currently unfeasible. Therefore, we specifically want to focus on finding 'tough word experts' for the low-pos*, low-neg* region of word space. Lower grain words (high-domall) should also be focused on since most NLP applications utilizing WSD (e.g. machine translation) require a coarser grain than practiced in Senseval evaluations investigated here. For such facilitated tasks, we recommend 'simple' classifiers trained with highly relevant and rich features, possibly fueled with extra knowledge (lexical or corporal).

As a further step, we plan to evaluate the more sophisticated ('complex') classifiers (Bagging, Boosting, RandomForests, Grading etc. [20]) and their different configurations (e.g. SVM's other kernel types than polynomial and RBF) to find out whether more maximally complementary systems can be found and more net gain obtained from 'specializing', reconfiguring or traditionally ensembling those classifiers. With those additions, we expect to be able to build a highly accurate 'ultimate OE', with maximally complementary custom systems based on performance profiles of base system types divided by the natural easiness/toughness boundary.

While it has been said that classifier performance cannot be predicted, we have presented findings at least for WSD, it seems there is hope still. We believe once a slightly better prediction accuracy is achieved (> 0.95), we can directly compare other systems to each other across datasets (e.g. Senseval-2 and Senseval-3) or even across tasks (e.g. text categorization [5] and word sense disambiguation [4,13] systems) and ultimately represent the regions of all classification systems (regardless of classification task, dataset and language) in terms of the case and system factors studied in this paper. This, however, requires further research across several fields involved with machine learning based classification. At the very least, we consider this paper to be helpful in understanding why WSD classification systems based on different classifiers and inputs can achieve equal results as seen in Sensevals [4,13].

References

1. Aha, D. W. Generalizing from case studies: A case study. In Proceedings of the Ninth International Conference on Machine Learning. Morgan Kaufmann. (1992)
2. Aha, D., Kibler, D. Instance-based learning algorithms. Machine Learning. 6:37-66. (1991)

3. Bay, S. D., and Pazzani, M. J. Characterizing model errors and differences. In 17th International Conference on Machine Learning (2000)
4. Edmonds, P., and Kilgarriff, A. Introduction to the Special Issue on evaluating word sense disambiguation programs. Journal of Natural Language Engineering 8(4) (2002).
5. Forman, G., and Cohen, I. Learning from Little: Comparison of Classifiers Given Little Training. ECML'04Learning from Little: Comparison of Classifiers Given Little Training. 15th European Conference on Machine Learning and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (2004)
6. Hoste, V., Hendrickx, I., Daelemans, W. and A. van den Bosch. Parameter optimization for machine-learning of word sense disambiguation. Journal of Natural Language Engineering, 8(4) (2002) 311-327.
7. John, G. and Langley, P. Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo (1995)
8. Legrand, S., Pulido JGR. A Hybrid Approach to Word Sense Disambiguation: Neural Clustering with Class Labeling. Knowledge Discovery and Ontologies workshop at 15th European Conference on Machine Learning (ECML) (2004).
9. Luo, F., Khan, L., Bastani F., Yen I-L and Zhou, J. A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles, Bioinformatics 2004 20(16):2605-2617, Oxford University Press. (2004)
10. Manning, C., Tolga Ilhan, H., Kamvar, S., Klein, D. and Toutanova, K. Combining Heterogeneous Classifiers for Word-Sense Disambiguation. Proceedings of SENSEVAL-2, Second International Workshop on Evaluating WSD Systems (2001) 87-90.
11. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. and Euler, T. YALE: Rapid Prototyping for Complex Data Mining Tasks. Proceedings of 12th ACM SIGKDD (2006)
12. Mihalcea, R. Word sense disambiguation with pattern learning and automatic feature selection. Journal of Natural Language Engineering, 8(4) (2002) 343-359.
13. Mihalcea, R., Kilgarriff, A. and Chklovski, T. The SENSEVAL-3 English lexical sample task. Proceedings of SENSEVAL-3 Workshop at ACL (2004).
14. Pedersen, T. Assessing System Agreement and Instance Difficulty in the Lexical Sample Tasks of SENSEVAL-2. Proceedings of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation (2002).
15. Pedersen, T. Machine learning with lexical features: The Duluth approach to Senseval. In Proceedings of the Senseval-2 Workshop (2001).
16. Quinlan, R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA. (1993)
17. Saarikoski, H. and Legrand. S. Building an Optimal WSD Ensemble Using Per-Word Selection of Best System. In Lecture Notes in Computer Science, Volume 4225/2006, Progress in Pattern Recognition, Image Analysis and Applications, CIARP (2006)
18. Seo, H-C., Rim, H-C. and Kim, S-H. KUNLP system in Senseval-3. Proceedings of SENSEVAL-2 Workshop (2001) 222-225.
19. Vapnik, V. N. The Nature of Statistical Learning Theory. Springer (1995)
20. Witten, I., Frank, E. Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann (2005).
21. Yarowsky, D., S. Cucerzan, R. Florian, C. Schafer and R. Wicentowski. The Johns Hopkins SENSEVAL2 System Descriptions. Proceedings of SENSEVAL-2 workshop (2002).
22. Yarowsky, D. and Florian, R. Evaluating sense disambiguation across diverse parameter spaces. Journal of Natural Language Engineering, 8(4) (2002) 293-311.