Combining Heterogeneous Classifiers for Word-Sense Disambiguation

Dan Klein, **Kristina Toutanova**, **H. Tolga Ilhan**,
Sepandar D. Kamvar and **Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305-9040, USA

Abstract

This paper discusses ensembles of simple but heterogeneous classifiers for word-sense disambiguation, examining the Stanford-CS224N system entered in the SENSEVAL-2 English lexical sample task. First-order classifiers are combined by a second-order classifier, which variously uses majority voting, weighted voting, or a maximum entropy model. While individual first-order classifiers perform comparably to middle-scoring teams' systems, the combination achieves high performance. We discuss trade-offs and empirical performance. Finally, we present an analysis of the combination, examining how ensemble performance depends on error independence and task difficulty.

1 Introduction

The problem of supervised word sense disambiguation (WSD) has been approached using many different classification algorithms, including naive-Bayes, decision trees, decision lists, and memory-based learners. While it is unquestionable that certain algorithms are better suited to the WSD problem than others (for a comparison, see Mooney (1996)), it seems that, given similar input features, various algorithms exhibit roughly similar accuracies.[1] This was supported by the SENSEVAL-2 results, where a large fraction of systems had scores clustered in a fairly narrow region (Senseval-2, 2001).

We began building our system with 23 supervised WSD systems, each submitted by a student taking the natural language processing course (CS224N) at Stanford University in Spring 2000. Students were free to implement whatever WSD method they chose. While most implemented variants of naive-Bayes, others implemented a range of other methods, including n-gram models, vector space models, and memory-based learners. Taken individually, the best of these systems would have turned in an accuracy of 61.2% in the SENSEVAL-2 English lexical sample task (which would have given it 6th place), while others would have produced middling to low performance. In this paper, we investigate how these classifiers behave in combination.

In section 2, we discuss the first-order classifiers and describe our methods of combination. In section 3, we discuss performance, analyzing what benefit was found from combination, and when. We also discuss aspects of the component systems which substantially influenced overall performance.

2 The System

2.1 Training Procedure

Figure 1 shows the high-level organization of our system. Individual first-order classifiers each map lists of context word tokens to word-sense predictions, and are self-contained WSD systems. The first-order classifiers are combined in a variety of ways with second-order classifiers. Second-order classifiers are selectors, taking a list of first-order out-

This paper is based on work supported in part by the National Science Foundation under Grants IIS-0085896 and IIS-9982226, by an NSF Graduate Fellowship, and by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University.

[1] In fact, we have observed that differences between implementations of a single classifier type, such as smoothing or window size, impacted accuracy far more than the choice of classification algorithm.

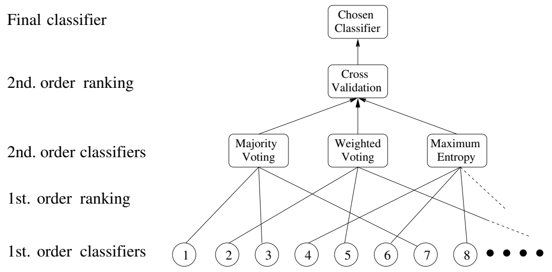

Figure 1: High-level system organization.

1	Split data into multiple training and held-out parts.
2	Rank first-order classifiers globally (across all words).
3	Rank first-order classifiers locally (per word), breaking ties with global ranks.
4	For each word w
5	For each size k
6	Choose the ensemble $E_{w,k}$ to be the top k classifiers
7	For each voting method m
8	Train the (k, m) second-order classifier with $E_{w,k}$
9	Rank the second-order classifier types (k, m) globally.
10	Rank the second-order classifier instances locally.
11	Choose the top-ranked second-order classifier for each word.
12	Retrain chosen per-word classifiers on entire training data.
13	Run these classifiers on test data, and evaluate results.

Table 1: The classifier construction process.

puts and choosing from among them. An outline of the classifier construction process is given in table 1. First, the training data was split into training and held-out sets for each word. This was done using 5 random bootstrap splits. Each split allocated 75% of the examples to training and 25% to held-out testing.[2] Held-out data was used both to select the subsets of first-order classifiers to be combined, and to select the combination methods.

For each word and each training split, the 23 first-order classifiers were (independently) trained and tested on held-out data. For each word, the first-order classifiers were ranked by their average performance on the held-out data, with the most accurate classifiers at the top of the rankings. Ties were broken by the classifiers' (weighted) average perfomance across all words.

For each word, we then constructed a set of candidate second-order classifiers. Second-order classifier types were identified by an ensemble size k and a combination method m. One instance of each second-order type was constructed for each word.

We originally considered ensemble sizes k in the range $\{1, 3, 5, 7, 9, 11, 13, 15\}$. For a second-order classifier with ensemble size k, the ensemble members were the top k first-order classifiers according to the local rank described above.

We combined first-order ensembles using one of three methods m:

- *Majority voting:* The sense output by the most first-order classifiers in the ensemble was chosen. Ties were broken by sense frequency, in favor of more frequent senses.
- *Weighted voting:* Each first-order classifier was assigned a voting weight (see below). The sense receiving the greatest total weighted vote was chosen.
- *Maximum entropy:* A maximum entropy classifier was trained (see below) and run on the outputs of the first-order classifiers.

We considered all pairs of k and m, and so for each word there were 24 possible second-order classifiers, though for $k = 1$ all three values of m are equivalent and were merged. The $k = 1$ ensemble, as well as the larger ensembles ($k \in \{9, 11, 13, 15\}$), did not help performance once we had good first-order classifier rankings (see section 3.4).

For m = *Majority*, there are no parameters to set. For the other two methods, we set the parameters of the (k, m) second-order classifier for a word w using the bootstrap splits of the training data for w.

In the same manner as for the first-order classifiers, we then ranked the second-order classifiers. For each word, there was the local ranking of the second-order classifiers, given by their (average) accuracy on held-out data. Ties in these rankings were broken by the average performance of the classifier type across all words. The top second-order classifier for each word was selected from these tie-broken rankings.

At this point, all first-order ensemble members and chosen second-order combination methods were retrained on the unsplit training data and run on the final test data.

[2]Bootstrap splits were used rather than standard n-fold cross-validation for two reasons. First, it allowed us to generate an arbitrary number of training/held-out pairs while still leaving substantial held-out data set sizes. Second, this approach is commonly used in the literature on ensembles. Its well-foundedness and theoretical properties are discussed in Breiman (1996). In retrospect, since we did not take proper advantage of the ability to generate numerous splits, it might have been just as well to use cross-validation.

It is important to stress that each target word was considered an entirely separate task, and different first- and second-order choices could be, and were, made for each word (see the discussion of table 2 below). Aggregate performance across words was only used for tie-breaking.

2.2 Combination Methods

Our second-order classifiers take training instances of the form $\bar{s} = (s, s_1, \ldots, s_k)$ where s is the correct sense and each s_i is the sense chosen by classifier i. All three of the combination schemes which we used can be seen as weighted voting, with different ways of estimating the voting weights λ_i of the first-order voters. In the simplest case, majority voting, we skip any attempt at statistical estimation and simply set each λ_i to be $1/k$.

For the method we actually call "weighted voting," we view the combination output as a mixture model in which each first-order system is a mixture component:

$$P(s|s_1, \ldots, s_k) = \sum_i \lambda_i P(s|s_i)$$

The conditional probabilties $P(s|s_i)$ assign mass one to the sense s_i chosen by classifier i. The mixture weights λ_i were estimated using EM to maximize the likelihood of the second-order training instances. In testing, the sense with the highest weighted vote, and hence highest posterior likelihood, is the selected sense.

For the maximum entropy classifier, we have a different model for the chosen sense s. In this case, it is an exponential model of the form:

$$P(s|s_1, \ldots, s_k) = \frac{\exp \sum_x \lambda_x f_x(s, s_1, \ldots, s_k)}{\sum_t \exp \sum_x \lambda_x f_x(t, s_1, \ldots, s_k)}$$

The features f_x are functions which are true over some subset of vectors $\bar{s}$. The original intent was to design features to recognize and exploit "sense expertise" in the individual classifiers. For example, one classifier might be trustworthy when reporting a certain sense but less so for other senses. However, there was not enough data to accurately estimate parameters for such models.[3] In fact, we noticed that, for certain words, simple majority voting performed better than the maximum entropy model. It also turned out that the most complex features we could get value from were features of the form:

$$f_i(s, s_1, \ldots, s_k) = 1 \iff s = s_i$$

That is, for each first-order classifier, there is a single feature which is true exactly when that classifier is correct. With only these features, the maximum entropy approach also reduces to a weighted vote; the s which maximizes the posterior probability $P(s|s_1, \ldots, s_k)$ also maximizes the vote:

$$v(s) = \textstyle\sum_i \lambda_i \delta(s_i = s)$$

The indicators δ are true for exactly one sense, and correspond to the simple f_i defined above.[4] The sense with the largest vote $v(s)$ will be the sense with the highest posterior probability $P(s|s_1, \ldots s_k)$ and will be chosen.

For the maximum entropy classifier, we estimate the weights by maximizing the likelihood of a held-out set, using the standard IIS algorithm (Berger et al., 1996). For both weighted schemes, we found that stopping the iterative procedures before convergence gave better results. IIS was halted after 50 rounds, while EM was halted after a single round. Both methods were initialized to uniform starting weights.

More importantly than changing the exact weight estimates, moving from method to method triggers broad qualitative changes in what kind of weights are allowed. With majority voting, classifiers all have equal, positive weights. With weighted voting, the weights are no longer required to be equal, but are still non-negative. With maximum entropy weighting, this non-negativity constraint is also relaxed, allowing classifiers' votes to actually reduce the score for the sense that classifier has chosen. Negative weights are in fact assigned quite frequently, and often seem to have the effect of using poor classifiers as "error masks" to cancel out common errors.

As we move from majority voting to weighted voting to maximum entropy, the estimation becomes

[3] The number of features was not large: only one for each (classifier, chosen sense, correct sense) triple. However, most senses are rarely chosen and rarely correct, so most features had zero or singleton support.

[4] If the ith classifier returns the correct sense s, then $\delta(s_i = s)$ is 1, otherwise it is zero.

more sophisticated, but also more prone to overfitting. Since solving the overfitting problem is hard, while choosing between classifiers based on held-out data is relatively easy, this spectrum gives us a way to gracefully handle the range of sparsities in the training corpora for different words.

2.3 Individual Classifiers

While our first-order classifiers implemented a variety of classification algorithms, the differences in their individual accuracies did not primarily stem from the algorithm chosen. Rather, implementation details led to the largest differences. For example, naive-Bayes classifiers which chose sensible window sizes, or dynamically chose between window sizes, tended to outperform those which chose poor sizes. Generally, the optimal windows were either of size one (for words with strong local syntactic or collocational cues) or of very large size (which detected more topical cues). Programs with hard-wired window sizes of, say, 5, performed poorly. Ironically, such middle-size windows were commonly chosen by students, but rarely useful; either extreme was a better design.[5]

Another implementation choice dramatically affecting performance of naive-Bayes systems was the amount and type of smoothing. Heavy smoothing and smoothing which backed off conditional distributions $P(w_j|s_i)$ to the relevant marginal $P(w_j)$ gave good results, while insufficient smoothing or backing off to uniform marginals gave substantially degraded results.[6]

There is one significant way in which our first-order classifiers were likely different from other teams' systems. In the original class project, students were guaranteed that the ambiguous word would appear only in a single orthographic form, and many of the systems depended on the input satisfying this guarantee. Since this was not true of the SENSEVAL-2 data, we mapped the ambiguous words (but not context words) to a citation form. We suspect that this lost quite a bit of information and negatively affected the system's overall performance, since there is considerable correlation between form and sense, especially for verbs. Nevertheless, we have made no attempt to re-engineer the student systems, and have not thoroughly investigated how big a difference this stemming made.

[5]Such window sizes were also apparently chosen by other SENSEVAL-2 systems, which commonly used "long distance" and "local" features, but defined local as a window size of 3–5 words on each side of the ambiguous word.

[6]In particular, there is a defective behavior with naive-Bayes where, when one smooths far too little, the chosen sense is the one which has occurred with the most words in the context window. For small training sets of skewed-prior data like the SENSEVAL-2 sets, this is invariably the common sense, regardless of the context words.

3 Results and Discussion

3.1 Results

Table 2 shows the results per word, and table 3 shows results by part-of-speech and overall, for the SENSEVAL-2 English lexical sample task. It also shows what second-order classifiers were selected for each word. 54.2% of the time, we made an optimal second-order classifier choice. When we chose wrong, we usually made a mistake in either ensemble size or method, rarely both. A wide range of second-order classifier types were chosen. As an overview of the benefit of combination, the globally best single classifier scored 61.2%, the locally best single classifier (best on test data) scored 62.2%, the globally best second order classifier (ME-7, best on test data) scored 63.2%, and our dynamic selection method scored 63.9%. Section 3.3 examines combination effectiveness more closely.

3.2 Changes from SENSEVAL-2

The system we originally submitted to the SENSEVAL-2 competition had an overall accuracy of 61.7%, putting it in 4th place in the revised rankings (among 21 supervised and 28 total systems). Assuming that our first-order classifiers were fixed black-boxes, we wanted an idea of how good our combination and selection methods were. To isolate the effectiveness of our second-order classifier choices, we compared our system to an oracle method (OR-BEST) which chose a word's second-order classifier based on test data (rather than held-out data). The overall accuracy of this oracle method was 65.4% at the time, a jump of 3.7%.[7] This gap was larger than the gap between the various top-scoring teams' systems. Therefore, while the test-set performance of the second-order classifiers is obviously not available, it was clear

[7]With other changes, OR-BEST rose to 66.1%.

	LB	Baselines		Combination			OR	UB	System	
Word	ALL	MFS	SNG	MJ-7	WT-7	ME-7	BEST	SOME	ACC	CL
art-n	28.6	41.8	50.6	52.0	54.1	52.0	58.2	69.4	58.2	WT-5
authority-n	45.7	33.7	61.3	69.6	69.6	65.2	69.6	78.3	66.3	WT-3
bar-n	31.1	39.7	63.7	61.6	69.5	72.2	72.2	81.5	72.2	ME-7
begin-v	50.0	58.6	70.0	83.6	84.3	88.2	88.2	94.6	83.6	MJ-7
blind-a	65.5	83.6	77.8	83.6	83.6	85.5	85.5	90.9	83.6	WT-7
bum-n	71.1	75.6	71.3	75.6	75.6	77.8	77.8	82.2	77.8	ME-7
call-v	1.5	25.8	33.3	25.8	30.3	27.3	34.8	62.1	30.3	WT-7
carry-v	9.1	22.7	27.8	34.8	33.3	33.3	37.9	62.1	33.3	MJ-5
chair-n	76.8	79.7	84.2	82.6	82.6	82.6	82.6	84.1	81.2	ME-3
channel-n	46.6	27.4	61.1	60.3	60.3	65.8	67.1	78.1	67.1	ME-3
child-n	34.4	54.7	57.9	67.2	70.3	70.3	75.0	90.6	71.9	WT-5
church-n	56.2	53.1	63.1	73.4	73.4	75.0	75.0	85.9	73.4	WT-7
circuit-n	52.9	27.1	70.9	65.9	65.9	78.8	78.8	80.0	78.8	ME-5
collaborate-v	90.0	90.0	92.9	90.0	90.0	90.0	90.0	90.0	90.0	WT-5
colorless-a	48.6	65.7	80.0	68.6	68.6	68.6	68.6	82.9	68.6	ME-5
cool-a	15.4	46.2	65.0	57.7	55.8	59.6	59.6	80.8	59.6	ME-5
day-n	36.6	59.3	58.4	69.0	68.3	66.2	69.0	82.8	63.4	WT-3
develop-v	11.6	29.0	35.2	42.0	43.5	42.0	43.5	68.1	42.0	MJ-3
draw-v	4.9	9.8	23.4	29.3	26.8	24.4	29.3	41.5	26.8	WT-5
dress-v	25.4	42.4	49.9	52.5	52.5	55.9	59.3	72.9	55.9	ME-7
drift-v	3.1	25.0	31.7	37.5	37.5	34.4	37.5	65.6	37.5	WT-5
drive-v	16.7	28.6	40.0	45.2	45.2	40.5	45.2	61.9	42.9	MJ-3
dyke-n	85.7	89.3	86.5	89.3	89.3	89.3	92.9	96.4	92.9	WT-3
face-v	82.8	83.9	80.9	83.9	83.9	82.8	83.9	84.9	83.9	WT-5
facility-n	36.2	48.3	70.5	67.2	70.7	65.5	74.1	86.2	70.7	WT-7
faithful-a	56.5	78.3	65.0	78.3	78.3	78.3	82.6	100.0	78.3	MJ-3
fatigue-n	67.4	76.7	83.9	88.4	90.7	90.7	90.7	93.0	90.7	MJ-5
feeling-n	29.4	56.9	76.7	62.7	70.6	72.5	74.5	86.3	72.5	WT-3
find-v	7.4	14.7	37.6	30.9	27.9	30.9	32.4	48.5	32.4	WT-3
fine-a	32.9	38.6	46.9	51.4	57.1	54.3	57.1	67.1	52.9	MJ-3
fit-a	51.7	51.7	87.7	89.7	89.7	86.2	93.1	96.6	93.1	MJ-5
free-a	26.8	39.0	58.2	65.9	65.9	61.0	65.9	74.4	64.6	ME-3
graceful-a	62.1	75.9	81.4	79.3	79.3	79.3	79.3	82.8	79.3	WT-5
green-a	69.1	78.7	80.0	83.0	83.0	83.0	85.1	88.3	83.0	MJ-3
grip-n	25.5	54.9	49.2	60.8	60.8	58.8	74.5	84.3	60.8	MJ-7
hearth-n	46.9	75.0	56.3	75.0	71.9	65.6	75.0	84.4	62.5	WT-3
holiday-n	77.4	83.9	89.7	83.9	83.9	80.6	83.9	87.1	83.9	WT-5
keep-v	19.4	37.3	36.1	38.8	49.3	52.2	52.2	65.7	52.2	WT-5
lady-n	60.4	69.8	67.7	75.5	75.5	77.4	77.4	81.1	75.5	WT-3
leave-v	21.2	31.8	29.1	43.9	53.0	50.0	54.5	68.2	54.5	WT-5
live-v	20.9	50.7	54.6	53.7	59.7	65.7	71.6	77.6	71.6	MJ-3
local-a	15.8	57.9	76.8	71.1	68.4	68.4	71.1	92.1	71.1	MJ-7
match-v	11.9	35.7	30.4	52.4	52.4	57.1	57.1	78.6	47.6	WT-3
material-n	39.1	42.0	56.0	55.1	55.1	50.7	66.7	73.9	66.7	WT-3
mouth-n	15.0	45.0	40.5	53.3	53.3	45.0	56.7	78.3	53.3	MJ-5
nation-n	70.3	70.3	71.1	70.3	70.3	70.3	70.3	70.3	70.3	WT-5
natural-a	18.4	27.2	50.4	49.5	50.5	58.3	58.3	76.7	55.3	WT-3
nature-n	23.9	45.7	51.3	63.0	67.4	65.2	67.4	82.6	60.9	MJ-5
oblique-a	51.7	69.0	73.7	82.8	82.8	82.8	86.2	89.7	79.3	WT-5
play-v	12.1	19.7	35.6	40.9	51.5	50.0	51.5	62.1	51.5	WT-5
post-n	26.6	31.6	66.5	49.4	57.0	65.8	67.1	73.4	67.1	ME-3
pull-v	1.7	21.7	27.7	21.7	21.7	28.3	28.3	46.7	23.3	WT-3
replace-v	28.9	53.3	49.0	57.8	53.3	60.0	60.0	77.8	57.8	MJ-7
restraint-n	35.6	31.1	53.9	71.1	68.9	71.1	71.1	82.2	66.7	ME-5
see-v	29.0	31.9	40.0	42.0	42.0	42.0	42.0	55.1	42.0	MJ-5
sense-n	18.9	22.6	46.3	64.2	60.4	50.9	64.2	79.2	64.2	MJ-7
serve-v	35.3	29.4	54.4	60.8	64.7	66.7	66.7	74.5	62.7	WT-5
simple-a	51.5	51.5	43.0	51.5	51.5	51.5	51.5	54.5	51.5	ME-3
solemn-a	96.0	96.0	89.2	96.0	96.0	96.0	96.0	96.0	96.0	WT-3
spade-n	66.7	63.6	81.8	75.8	75.8	78.8	78.8	81.8	78.8	WT-3
stress-n	7.7	46.2	47.0	43.6	43.6	35.9	51.3	82.1	48.7	WT-5
strike-v	5.6	16.7	32.3	31.5	29.6	29.6	40.7	55.6	31.5	MJ-5
train-v	22.2	30.2	48.3	57.1	57.1	54.0	57.1	76.2	57.1	WT-7
treat-v	36.4	38.6	51.8	54.5	54.5	52.3	54.5	70.5	52.3	WT-3
turn-v	1.5	14.9	38.8	32.8	29.9	32.8	35.8	52.2	31.3	MJ-5
use-v	61.8	65.8	69.6	65.8	65.8	72.4	72.4	75.0	72.4	ME-3
vital-a	84.2	92.1	91.5	92.1	92.1	92.1	92.1	92.1	92.1	WT-5
wander-v	70.0	80.0	83.2	80.0	82.0	82.0	82.0	84.0	80.0	ME-3
wash-v	16.7	25.0	40.0	58.3	58.3	25.0	58.3	83.3	58.3	MJ-7
work-v	10.0	26.7	28.1	43.3	43.3	41.7	45.0	63.3	45.0	WT-3
yew-n	75.0	78.6	81.4	78.6	78.6	78.6	78.6	82.1	78.6	WT-5

Table 2: Results by word for the SENSEVAL-2 English lexical sample task. Lower bound (LB): ALL is how often all of the first-orders chose correctly. Baselines (BL): MFS is the most-frequent-sense baseline, SNG is the best single first-order classifier as chosen on held-out data for that word. Fixed combinations: majority vote (MJ), weighted vote (WT), maximum entropy (ME). Oracle bound (OR): BEST is the best second-order classifier as measured on the test data. Upper bound (UB): SOME is how often at least one first-order classifier produced the correct answer. Methods which are ensemble-size dependent are shown for $k = 7$. System choices: ACC is the accuracy of the selection the system makes based on held-out data. CL is the 2nd-order classifier selected.

that a more sophisticated or better-tuned method of selecting combination models could lead to significant improvement. In fact, changing only ranking methods, which are discussed further in the next section, resulted in an increase in final accuracy for our system to the current score of 63.9%, which would have placed it 1st in the SENSEVAL-2 preliminary results or 2nd in the revised results. Our

	LB	Baselines		Combination			OR	UB	System
	ALL	MFS	SNG	MJ-7	WT-7	ME-7	BEST	SOME	ACC
noun	42.5	50.5	63.8	66.4	67.9	67.8	71.9	81.2	69.7
adj.	45.1	57.8	66.7	69.0	69.4	69.9	71.6	81.0	69.9
verb	28.8	40.2	48.7	53.4	54.7	55.8	58.2	71.2	55.7
avg.	46.5	47.5	62.2	61.5	62.7	63.2	68.9	72.0	63.9

Table 3: Results by part-of-speech, and overall.

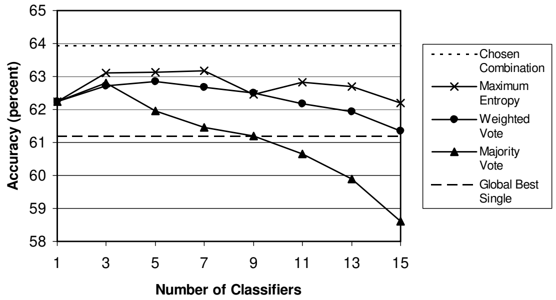

Figure 2: Accuracy of the various combination methods as the ensemble size varies. The three combination methods are shown. In addition, the *globally best single classifier* is the single first-order classifier with the highest overall accuracy on the test data. *Chosen combination* is our final system's score. These two are both independent of k in this graph.

final accuracy is thus higher than the first draft of the system, and, in particular, the classifier selection gap between actual performance and the OR-BEST oracle has been substantially decreased.

In addition, since the top first-order classifiers were more reliably identified, larger ensembles were no longer beneficial in the revised system, for an interesting reason. When the first-order rankings were poorly estimated, large ensembles and weighted methods were important for achieving good accuracy, because the weighting scheme could "rescue" good classifiers which had been incorrectly ranked low. In our current system, however, first-order classifiers were ranked reliably enough that we could restrict our ensemble sizes to $k \in \{1, 3, 5, 7\}$. Furthermore, since $k = 1$ was only chosen a few times, usually among ties, we removed that option as well.

3.3 Combination Methods and Ensemble Size

Our system differs from the typical ensemble of classifiers in that the first-order classifiers are not merely perturbations of each other, but are highly varied in both quality and character. This scenario has been investigated before, e.g. (Zhang et al., 1992), but is not the common case. With such heterogeneity, having more classifiers is not always better. Figure 2 shows how the three combination methods' average scores varied with the number of component classifiers used. Initially, accuracy increases as added classifiers bring value to the ensemble. However, as lower-quality classifiers are added in, the better classifiers are steadily drowned out. The weighted vote and maximum entropy combinations are much less affected by low-quality classifiers than the majority vote, being able to suppress them with low weights. Still, majority vote over small ensembles was effective for some words where weights could not be usefully set by the other methods.

3.4 Ranking Methods

Because of the effects described above, it was necessary to identify which classifiers were worth including for a given word. A global ranking of first-order classifiers, averaged over all words, was not effective because the strengths of the classifiers were so different. In fact, every single first-order classifier was a top-5 performer on at least one word. On the other hand, SENSEVAL-2 training sets were often very small, and very skewed towards a frequent most-frequent-sense. As a result, accuracy estimates based on single words' held-out data produced frequent ties. The average size of the per-word largest set of tied first-order classifiers was 3.6 (with a maximum of 23 on the word *collaborate* where all tied). The second-order local rankings also produced many ties. For the top position (the most important for second-order ranks) 43.1% of the words had local ties.

In our submitted entry, all ties were broken unintelligently (in an arbitrary manner based on the order in which systems were listed in a file). The approach of local ranking with global tie-breaking presented in this paper was much more successful according to two distinct measures. First, it predicted the true ranks more accurately, (measured by the Spearman rank correlation: 0.08 for global ranks, 0.63 for globally-broken local ranks) and gave better final accuracy scores (63.5% with global, 63.9% with globally-broken local – significant only at p=0.1 by a sign test at the word type level).

The other ranking that our system attempts to estimate is the per-word ranking of the second-order classifiers. In this case, however, we are only ever concerned with which classifier ends up being ranked first, as only that classifier is chosen. Again, globally-broken local ranks were the most effective, choosing a second-order classifier which was actually top-performing on test data for 54% of the words, as opposed to 50% for global selection (and increasing the overall accuracy from 62.8% to 63.9% – significant at p=0.01, sign test).

These results stress that ranking, and effective tie-breaking, are important for a system such as ours where the classifiers are so divergent in behavior.

3.5 Combination

When combining classifiers, one would like to know when and how the combination will outperform the individuals. One factor (Tumer and Ghosh, 1996) is how complementary the mistakes of the individual classifiers are. If all make the same mistakes, combination can do no good. We can measure this complementarity by averaging, over all pairs of first-order classifiers, the fraction of errors that pair has in common. This gives average pairwise error independence. Another factor is the difficulty of the word being disambiguated. A high most-frequent-sense baseline (BL-MFS) means that there is little room for improvement by combining classifiers. Figure 3 shows, for the global top 7 first-order classifiers, the absolute gain between their average accuracy (BL-AVG-7) and the accuracy of their majority combination (MJ-7). The quantity on the x-axis is the difference between the pairwise independence and the baseline accuracy. The pattern is loose, but clear. When either independence increases or the word's difficulty (as indicated by the BL-MFS baseline) increases, the combination tends to win by a greater amount.

Figure 4 shows how the average pairwise independent error fraction (api) varies as we add classifiers. Here classifiers are added in an order based on their accuracy on the entire test set. For each k, the average is over all pairs of classifiers in the top k and all samples of all words. This graph should be compared to figure 2. After the third classifier, adding classifiers reduces the api, and the performance of the majority vote begins to drop at exactly this point. However, the weighted methods continue to gain in accuracy since they have the capacity to downweight classifiers which hurt held-out accuracy.

The drop in api reflects that the newly added systems are no longer bringing many new correct answers to the collection. However, they can still add

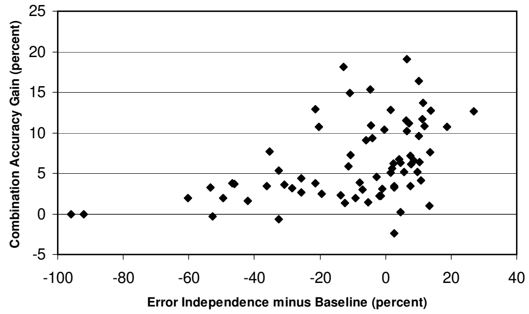

Figure 3: Gain in accuracy of majority vote over the average component performance as (pairwise independence − baseline accuracy) grows.

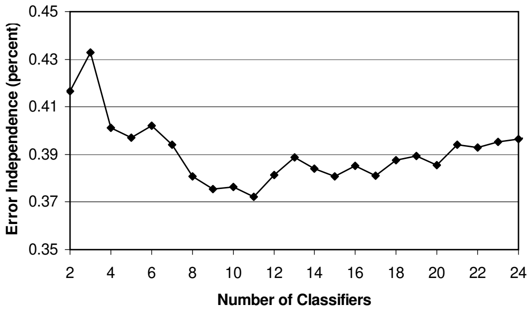

Figure 4: The average pairwise error independence of classifiers as their number is increased.

deciding votes in areas where the ensemble had the right answer, but did not choose it. The final gradual rise in api reflects the somewhat patternless new errors that substantially lower-performing systems unfortunately bring to the ensemble.

4 Conclusions

In this paper, we have explored ensemble sizes, combination methods, bounds for what can be expected from combinations, factors in the performance of individual classifiers, and methods of improving performance by effective tie-breaking. In accord with much recent work on classifier combination, e.g. (Breiman, 1996; Bauer and Kohavi, 1999), we have demonstrated that the combination of classifiers can lead to a substantial performance increase over the individual classifiers within the domain of WSD. In addition, we have shown that highly varying component systems augment each other well and that adding lower-scoring systems can still improve ensemble performance, at least to a certain point. A particular emphasis of our research has been how to make the combination robust to both the wide range of first-order classifier accuracies and to the sparsity of the available training data. Careful but greedy determination of rankings proved to be effective, capturing the highly word-dependent strengths of our classifiers. The resulting system's overall accuracy is very high, despite the medium level of accuracy of the component systems.

5 Acknowledgments

We would like to thank the following people for contributing their classifiers to the Stanford CS224N system: Zoe Abrams, Jenny Berglund, Dmitri Bobrovnikoff, Chris Callison-Burch, Marcos Chavira, Shipra Dingare, Elizabeth Douglas, Sarah Harris, Ido Milstein, Jyotirmoy Paul, Soumya Raychaudhuri, Paul Ruhlen, Magnus Sandberg, Adil Sherwani, Philip Shilane, Joshua Solomin, Patrick Sutphin, Yuliya Tarnikova, Ben Taskar, Kristina Toutanova, Christopher Unkel, and Vincent Vanhoucke.

References

Alan Agresti. 1990. *Categorical Data Analysis*. John Wiley & Sons.

Eric Bauer and Ron Kohavi. 1999. An empirical comparison of voting classication algorithms: Bagging, boosting and variants. *Machine Learning*, 36:105–139.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24:123–140.

Raymond J. Mooney. 1996. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *EMNLP 1*.

Senseval-2. 2001. Senseval-2 proceedings, in publication.

Kagan Tumer and Joydeep Ghosh. 1996. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8:385–404.

Xiru Zhang, Jill Mesirov, and David L. Waltz. 1992. Hybrid system for protein structure prediction. *Journal of Molecular Biology*, 225:1049–1063.