N Semantic Classes are Harder than Two

Ben Carterette[*]
CIIR
University of Massachusetts
Amherst, MA 01003
carteret@cs.umass.edu

Rosie Jones
Yahoo! Research
3333 Empire Ave.
Burbank, CA 91504
jonesr@yahoo-inc.com

Wiley Greiner[*]
Los Angeles Software Inc.
1329 Pine Street
Santa Monica, CA 90405
w.greiner@lasoft.com

Cory Barr
Yahoo! Research
3333 Empire Ave.
Burbank, CA 91504
barrc@yahoo-inc.com

Abstract

We show that we can automatically classify semantically related phrases into 10 classes. Classification robustness is improved by training with multiple sources of evidence, including within-document cooccurrence, HTML markup, syntactic relationships in sentences, substitutability in query logs, and string similarity. Our work provides a benchmark for automatic *n-way* classification into WordNet's semantic classes, both on a TREC news corpus and on a corpus of substitutable search query phrases.

1 Introduction

Identifying semantically related phrases has been demonstrated to be useful in information retrieval (Anick, 2003; Terra and Clarke, 2004) and sponsored search (Jones et al., 2006). Work on semantic entailment often includes lexical entailment as a subtask (Dagan et al., 2005).

We draw a distinction between the task of identifying terms which are topically related and identifying the specific semantic class. For example, the terms "dog", "puppy", "canine", "schnauzer", "cat" and "pet" are highly related terms, which can be identified using techniques that include distributional similarity (Lee, 1999) and within-document cooccurrence measures such as pointwise mutual information (Turney et al., 2003). These techniques, however, do not allow us to distinguish the more specific relationships:

- hypernym(dog,puppy)
- hyponym(dog,canine)
- coordinate(dog,cat)

[*]This work was carried out while these authors were at Yahoo! Research.

Lexical resources such as WordNet (Miller, 1995) are extremely useful, but are limited by being manually constructed. They do not contain semantic class relationships for the many new terms we encounter in text such as web documents, for example "mp3 player" or "ipod". We can use WordNet as training data for such classification to the extent that the training on pairs found in WordNet and testing on pairs found outside WordNet provides accurate generalization.

We describe a set of features used to train n-way supervised machine-learned classification of semantic classes for arbitrary pairs of phrases. Redundancy in the sources of our feature information means that we are able to provide coverage over an extremely large vocabulary of phrases. We contrast this with techniques that require parsing of natural language sentences (Snow et al., 2005) which, while providing reasonable performance, can only be applied to a restricted vocabulary of phrases cooccuring in sentences.

Our contributions are:

- Demonstration that binary classification removes the difficult cases of classification into closely related semantic classes
- Demonstration that dependency parser paths are inadequate for semantic classification into 7 WordNet classes on TREC news corpora
- A benchmark of 10-class semantic classification over highly substitutable query phrases
- Demonstration that training a classifier using WordNet for labeling does not generalize well to query pairs
- Demonstration that much of the performance in classification can be attained using only

syntactic features

- A learning curve for classification of query phrase pairs that suggests the primary bottleneck is manually labeled training instances: we expect our benchmark to be surpassed.

2 Relation to Previous Work

Snow et al. (2005) demonstrated binary classification of hypernyms and non-hypernyms using WordNet (Miller, 1995) as a source of training labels. Using dependency parse tree paths as features, they were able to generalize from WordNet labelings to human labelings.

Turney et al. (2003) combined features to answer multiple-choice synonym questions from the TOEFL test and verbal analogy questions from the SAT college entrance exam. The multiple-choice questions typically do not consist of multiple closely related terms. A typical example is given by Turney:

• hidden::	(a) laughable	(c) ancient
	(b) veiled	(d) revealed

Note that only (b) and (d) are at all related to the term, so the algorithm only needs to distinguish antonyms from synonyms, not synonyms from say hypernyms.

We use as input phrase pairs recorded in query logs that web searchers substitute during search sessions. We find much more closely related phrases:

	(a) secret	(e) hiden
• hidden::	(b) hidden camera	(f) voyeur
	(c) hidden cam	(g) hide
	(d) spy	

This set contains a context-dependent synonym, topically related verbs and nouns, and a spelling correction. All of these could cooccur on web pages, so simple cooccurrence statistics may not be sufficient to classify each according to the semantic type.

We show that the techniques used to perform binary semantic classification do not work as well when extended to a full n-way semantic classification. We show that using a variety of features performs better than any feature alone.

3 Identifying Candidate Phrases for Classification

In this section we introduce the two data sources we use to extract sets of candidate related phrases for classification: a TREC-WordNet intersection and query logs.

3.1 Noun-Phrase Pairs Cooccuring in TREC News Sentences

The first is a data-set derived from TREC news corpora and WordNet used in previous work for binary semantic class classification (Snow et al., 2005). We extract two sets of candidate-related pairs from these corpora, one restricted and one more complete set.

Snow et al. obtained training data from the intersection of noun-phrases cooccuring in sentences in a TREC news corpus and those that can be labeled unambiguously as hypernyms or non-hypernyms using WordNet. We use a restricted set since instances selected in the previous work are a subset of the instances one is likely to encounter in text. The pairs are generally either related in one type of relationship, or completely unrelated.

In general we may be able to identify related phrases (for example with distributional similarity (Lee, 1999)), but would like to be able to automatically classify the related phrases by the type of the relationship. For this task we identify a larger set of candidate-related phrases.

3.2 Query Log Data

To find phrases that are similar or substitutable for web searchers, we turn to logs of user search sessions. We look at *query reformulations*: a pair of successive queries issued by a single user on a single day. We collapse repeated searches for the same terms, as well as query pair sequences repeated by the same user on the same day.

3.2.1 Substitutable Query Segments

Whole queries tend to consist of several concepts together, for example "new york | maps" or "britney spears | mp3s". We identify segments or phrases using a measure over adjacent terms similar to mutual information. Substitutions occur at the level of segments. For example, a user may initially search for "britney spears | mp3s", then search for "britney spears | music". By aligning query pairs with a single substituted segment, we generate pairs of phrases which a user has substituted. In this example, the phrase "mp3s" was substituted by the phrase "music".

Aggregating substitutable pairs over millions of users and millions of search sessions, we can calculate the probability of each such rewrite, then test each pair for statistical significance to eliminate phrase rewrites which occurred in a small number of sessions, perhaps by chance. To test for statistical significance we use the pair independence likelihood ratio, or log-likelihood ratio, test. This metric tests the hypothesis that the probability of phrase β is the same whether phrase α has been seen or not by calculating the likelihood of the observed data under a binomial distribution using probabilities derived using each hypothesis (Dunning, 1993).

$$log\lambda = log\frac{L\left(P(\beta|\alpha) = P(\beta|\neg\alpha)\right)}{L\left(P(\beta|\alpha) \neq P(\beta|\neg\alpha)\right)}$$

A high negative value for λ suggests a strong dependence between query α and query β.

4 Labeling Phrase Pairs for Supervised Learning

We took a random sample of query segment substitutions from our query logs to be labeled. The sampling was limited to pairs that were frequent substitutions for each other to ensure a high probability of the segments having some relationship.

4.1 WordNet Labeling

WordNet is a large lexical database of English words. In addition to defining several hundred thousand words, it defines *synonym sets*, or *synsets*, of words that represent some underlying lexical concept, plus relationships between synsets. The most frequent relationships between noun-phrases are *synonym, hyponym, hypernym,* and *coordinate*, defined in Table 1. We also may use *meronym* and *holonym,* defined as the PART-OF relationship.

We used WordNet to automatically label the subset of our sample for which both phrases occur in WordNet. Any sense of the first segment having a relationship to any sense of the second would result in the pair being labeled. Since WordNet contains many other relationships in addition to those listed above, we group the rest into the *other* category. If the segments had no relationship in WordNet, they were labeled *no relationship*.

4.2 Segment Pair Labels

Phrase pairs passing a statistical test are common reformulations, but can be of many semantic types. Rieh and Xie (2001) categorized types of query reformulations, defining 10 general categories: *specification, generalization, synonym, parallel movement, term variations, operator usage, error correction, general resource, special resource,* and *site URLs*. We redefine these slightly to apply to query segments. The summary of the definitions is shown in Table 1, along with the distribution in the data of pairs passing the statistical test.

4.2.1 Hand Labeling

More than 90% of phrases in query logs do not appear in WordNet due to being spelling errors, web site URLs, proper nouns of a temporal nature, etc. Six annotators labeled $2,463$ segment pairs selected randomly from our sample. Annotators agreed on the label of 78% of pairs, with a Kappa statistic of $.74$.

5 Automatic Classification

We wish to perform supervised classification of pairs of phrases into semantic classes. To do this, we will assign features to each pair of phrases, which may be predictive of their semantic relationship, then use a machine-learned classifier to assign weights to these features. In Section 7 we will look at the learned weights and discuss which features are most significant for identifying which semantic classes.

5.1 Features

Features for query substitution pairs are extracted from query logs and web pages.

5.1.1 Web Page / Document Features

We submit the two segments to a web search engine as a conjunctive query and download the top 50 results. Each result is converted into an HTML Document Object Model (DOM) tree and segmented into sentences.

Dependency Tree Paths The path from the first segment to the second in a dependency parse tree generated by MINIPAR (Lin, 1998) from sentences in which both segments appear. These were previously used by Snow et al. (2005). These features were extracted from web pages in all experiments, except where we identify that we used TREC news stories (the same data as used by Snow et al.).

HTML Paths The paths from DOM tree nodes the first segment appears in to nodes the second segment appears in. The value is the number of times the path occurs with the pair.

Class	Description	Example	%
synonym	one phrase can be used in place of the other without loss in meaning	*low cost; cheap*	4.2
hypernym	X is a hypernym of Y if and only if Y is a X	*muscle car; mustang*	2.0
hyponym	X is a hyponym of Y if and only if X is a Y (inverse of hypernymy)	*lotus; flowers*	2.0
coordinate	there is some Z such that X and Y are both Zs	*aquarius; gemini*	13.9
generalization	X is a generalization of Y if X contains less information about the topic	*lyrics; santana lyrics*	4.8
specialization	X is a specification of Y if X contains more information about the topic	*credit card; card*	4.7
spelling change	spelling errors, typos, punctuation changes, spacing changes	*peopl; people*	14.9
stemmed form	X and Y have the same lemmas	*ant; ants*	3.4
URL change	X and Y are related and X or Y is a URL	*alliance; alliance.com*	29.8
other relationship	X and Y are related in some other way	*flagpoles; flags*	9.8
no relationship	X and Y are not related in any obvious way	*crypt; tree*	10.4

Table 1: Semantic relationships between phrases rewritten in query reformulation sessions, along with their prevalence in our data.

Lexico-syntactic Patterns (Hearst, 1992) A substring occurring between the two segments extracted from text in nodes in which both segments appear. In the example fragment "authors such as Shakespeare", the feature is "such as" and the value is the number of times the substring appears between "author" and "Shakespeare".

5.1.2 Query Pair Features

Table 2 summarizes features that are induced from the query strings themselves or calculated from query log data.

5.2 Additional Training Pairs

We can double our training set by adding for each pair u_1, u_2 a new pair u_2, u_1. The class of the new pair is the same as the old in all cases but *hypernym, hyponym, specification,* and *generalization,* which are inverted. Features are reversed from $f(u_1, u_2)$ to $f(u_2, u_1)$.

A pair and its inverse have different sets of features, so splitting the set randomly into training and testing sets should not result in resubstitution error. Nonetheless, we ensure that a pair and its inverse are not separated for training and testing.

5.3 Classifier

For each class we train a binary one-vs.-all linear-kernel support vector machine (SVM) using the optimization algorithm of Keerthi and DeCoste (2005).

5.3.1 Meta-Classifier

For n-class classification, we calibrate SVM scores to probabilities using the method described by Platt (2000). This gives us $P(class|pair)$ for each pair. The final classification for a pair is $argmax_{class} P(class|pair)$.

Source	Snow (NIPS 2005)	Experiment
Task	binary hypernym	binary hypernym
Data	WordNet-TREC	WordNet-TREC
Instance Count	752,311	752,311
Features	minipar paths	minipar paths
Feature Count	69,592	69,592
Classifier	logistic Regression	linear SVM
maxF	0.348	**0.453**

Table 3: Snow et al's (2005) reported performance using linear regression, and our reproduction of the same experiment, using a support vector machine (SVM).

5.3.2 Evaluation

Binary classifiers are evaluated by ranking instances by classification score and finding the Max F1 (the harmonic mean of precision and recall; ranges from 0 to 1) and area under the ROC curve (AUC; ranges from 0.5 to 1 with at least 0.8 being "good"). The meta-classifier is evaluated by precision and recall of each class and classification accuracy of all instances.

6 Experiments

6.1 Baseline Comparison to Snow et al.'s Previous Hypernym Classification on WordNet-TREC data

Snow et al. (2005) evaluated binary classification of noun-phrase pairs as *hypernyms* or *non-hypernyms*. When training and testing on WordNet-labeled pairs from TREC sentences, they report classifier Max F of 0.348, using dependency path features and logistic regression. To justify our choice of an SVM for classification, we replicated their work. Snow et al. provided us with their data. With our SVM we achieved a Max F of 0.453, 30% higher than they reported.

6.2 Extending Snow et al.'s WordNet-TREC Binary Classification to N Classes

Snow et al. select pairs that are "Known Hypernyms" (the first sense of the first word is a hy-

Feature	Description
Levenshtein Distance	# character insertions/deletions/substitutions to change query α to query β (Levenshtein, 1966).
Word Overlap Percent	# words the two queries have in common, divided by num. words in the longer query.
Possible Stem	1 if the two segments stem to the same root using the Porter stemmer.
Substring Containment	1 if the first segment is a substring of the second.
Is URL	1 if either segment matches a handmade URL regexp.
Query Pair Frequency	# times the pair was seen in the entire unlabeled corpus of query pairs.
Log Likelihood Ratio	The Log Likelihood Ratio described in Section 3.2.1 Formula 3.2.1
Dice and Jaccard Coefficients	Measures of the similarity of substitutes for and by the two phrases.

Table 2: Syntactic and statistical features over pairs of phrases.

ponym of the first sense of the second and both have no more than one tagged sense in the Brown corpus) and "Known Non-Hypernyms" (no sense of the first word is a hyponym of any sense of the second). We wished to test whether making the classes less cleanly separable would affect the results, and also whether we could use these features for n-way classification.

From the same TREC corpus we extracted *known synonym*, *known hyponym*, *known coordinate*, *known meronym*, and *known holonym* pairs. Each of these classes is defined analogously to the *known hypernym* class; we selected these six relationships because they are the six most common. A pair is labeled *known no-relationship* if no sense of the first word has any relationship to any sense of the second word. The class distribution was selected to match as closely as possible that observed in query logs. We labeled 50,000 pairs total.

Results are shown in Table 4(a). Although AUC is fairly high for all classes, MaxF is low for all but two. MaxF has degraded quite a bit for hypernyms from Table 3. Removing all instances except hypernym and no relationship brings MaxF up to 0.45, suggesting that the additional classes make it harder to separate hypernyms.

Metaclassifier accuracy is very good, but this is due to high recall of *no relationship* and *coordinate* pairs: more than 80% of instances with some relationship are predicted to be coordinates, and most of the rest are predicted no relationship. It seems that we are only distinguishing between *no* vs. *some* relationship.

The size of the *no relationship* class may be biasing the results. We removed those instances, but performance of the n-class classifier did not improve (Table 4(b)). MaxF of binary classifiers did improve, even though AUC is much worse.

6.3 N-Class Classification of Query Pairs

We now use query pairs rather than TREC pairs.

6.3.1 Classification Using Only Dependency Paths

We first limit features to dependency paths in order to compare to the prior results. Dependency paths cannot be obtained for all query phrase pairs, since the two phrases must appear in the same sentence together. We used only the pairs for which we could get path features, about 32% of the total.

Table 5(a) shows results of binary classification and metaclassification on those instances using dependency path features only. We can see that dependency paths do not perform very well on their own: most instances are assigned to the "coordinate" class that comprises a plurality of instances.

A comparison of Tables 5(a) and 4(a) suggests that classifying query substitution pairs is harder than classifying TREC phrases.

Table 5(b) shows the results of binary classification and metaclassification on the same instances using all features. Using all features improves performance dramatically on each individual binary classifier as well as the metaclassifier.

6.3.2 Classification on All Query Pairs Using All Features

We now expand to all of our hand-labeled pairs. Table 6(a) shows results of binary and meta classification; Figure 1 shows precision-recall curves for 10 binary classifiers (excluding URLs). Our classifier does quite well on every class but hypernym and hyponym. These two make up a very small percentage of the data, so it is not surprising that performance would be so poor.

The metaclassifier achieved 71% accuracy. This is significantly better than random or majority-class baselines, and close to our 78% interannotator agreement. Thresholding the metaclassifier to pairs with greater than .5 max class probability (68% of instances) gives 85% accuracy.

Next we wish to see how much of the performance can be maintained without using the com-

	binary		*n*-way		data
class	maxF	AUC	prec	rec	%
no rel	.980	.986	.979	.985	80.0
synonym	.028	**.856**	0	0	0.3
hypernym	.185	**.888**	.512	.019	2.1
hyponym	.193	**.890**	.462	.016	2.1
coordinate	.808	**.971**	.714	.931	14.8
meronym	.158	**.905**	**.615**	.050	0.3
holonym	.120	**.883**	**.909**	.062	0.3
metaclassifier accuracy			**.927**		

(a) All seven WordNet classes. The high accuracy is mostly due to high recall of *no rel* and *coordinate* classes.

binary		*n*-way		data
maxF	AUC	prec	rec	%
–	–	–	–	0
.086	.683	0	0	1.7
.337	.708	**.563**	**.077**	10.6
.341	.720	**.527**	**.080**	10.6
.857	.737	**.757**	**.986**	74.1
.251	.777	.500	**.068**	1.5
.277	.767	.522	**.075**	1.5
–		.749		

(b) Removing *no relationship* instances improves MaxF and recall of all classes, but performance is generally worse.

Table 4: Performance of 7 binary classifier and metaclassifiers on phrase-pairs cooccuring in TREC data labeled with WordNet classes, using minipar dependency features. These features do not seem to be adequate for distinguishing classes other than *coordinate* and *no-relationship*.

	binary		*n*-way	
class	maxf	auc	prec	rec
no rel	.281	.611	.067	.006
synonym	.269	.656	.293	.167
hypernym	.140	.626	0	0
hyponym	.121	.610	0	0
coordinate	.506	.760	.303	**.888**
spelling	.288	.677	.121	.022
stemmed	.571	.834	.769	.260
URL	.742	.919	.767	.691
generalization	.082	.547	0	0
specification	.085	.528	0	0
other	.393	.681	.384	.364
metaclassifier accuracy			.385	

(a) Dependency tree paths only.

binary		*n*-way		data	
maxf	auc	prec	rec	%	% full
.602	**.883**	**.639**	**.497**	10.6	3.5
.477	**.851**	**.571**	**.278**	4.5	1.5
.167	**.686**	**.125**	**.017**	3.7	1.2
.136	**.660**	0	0	3.7	1.2
.747	**.935**	**.624**	.862	21.0	6.9
.814	**.970**	**.703**	**.916**	11.0	3.6
.781	**.972**	**.788**	**.675**	4.8	1.6
1	**1**	**1**	**1**	16.2	5.3
.490	**.883**	**.489**	**.393**	3.5	1.1
.584	**.854**	**.600**	**.589**	3.5	1.1
.641	**.895**	**.603**	**.661**	17.5	5.7
–		.692		—	

(b) All features.

Table 5: Binary and metaclassifier performance on the 32% of hand-labeled instances with dependency path features. Adding all our features significantly improves performance over just using dependency paths.

putationally expensive syntactic parsing of dependency paths. To estimate the marginal gain of the other features over the dependency paths, we excluded the latter features and retrained our classifiers. Results are shown in Table 6(b). Even though binary and meta-classifier performance decreases on all classes but generalizations and specifications, much of the performance is maintained.

Because URL changes are easily identifiable by the **IsURL** feature, we removed those instances and retrained the classifiers. Results are shown in Table 6(c). Although overall accuracy is worse, individual class performance is still high, allowing us to conclude our results are not only due to the ease of classifying URLs.

We generated a learning curve by randomly sampling instances, training the binary classifiers on that subset, and training the metaclassifier on the results of the binary classifiers. The curve is shown in Figure 2. With 10% of the instances, we have a metaclassifier accuracy of 59%; with 100% of the data, accuracy is 71%. Accuracy shows no sign of falling off with more instances.

6.4 Training on WordNet-Labeled Pairs Only

Figure 2 implies that more labeled instances will lead to greater accuracy. However, manually labeled instances are generally expensive to obtain. Here we look to other sources of labeled instances for additional training pairs.

6.4.1 Training and Testing on WordNet

We trained and tested five classifiers using 10-fold cross validation on our set of WordNet-labeled query segment pairs. Results for each class are shown in Table 7. We seem to have regressed to predicting *no* vs. *some* relationship.

Because these results are not as good as the human-labeled results, we believe that some of our performance must be due to peculiarities of our data. That is not unexpected: since words that appear in WordNet are very common, features are much noisier than features associated with query entities that are often structured within web pages.

	binary		*n*-way		binary		*n*-way		data	binary		*n*-way	
class	maxf	auc	prec	rec	maxf	auc	prec	rec	%	maxf	auc	prec	rec
no rel	**.531**	**.878**	**.616**	**.643**	.466	.764	.549	.482	10.4	.512	.808	.502	.486
synonym	**.355**	**.820**	**.506**	**.212**	.351	.745	.493	.178	4.2	.350	.759	.478	.212
hypernym	**.173**	**.821**	.100	**.020**	.133	.728	0	0	2.0	.156	.710	**.250**	.020
hyponym	.173	**.797**	.059	.010	.163	.733	0	0	2.0	**.187**	.739	**.125**	**.020**
coordinate	**.635**	**.921**	**.590**	.703	.539	.832	.565	.732	13.9	.634	.885	.587	**.706**
spelling	**.778**	**.960**	.625	.904	.723	.917	**.628**	.902	14.9	.774	.939	.617	**.906**
stemmed	.703	**.973**	.786	.589	.656	.964	.797	.583	3.4	**.717**	.967	**.802**	**.601**
URL	1	1	1	1	1	1	1	1	29.8	–	–	–	–
generalization	.565	**.916**	.575	.483	.492	.852	**.604**	.604	4.8	**.581**	.885	.598	**.634**
specification	.661	**.926**	.652	.506	.578	.869	**.670**	**.644**	4.7	**.665**	.906	.657	.468
other	**.539**	**.898**	**.575**	**.483**	.436	.790	.550	.444	9.8	.529	.847	.559	.469
metaclassifier accuracy			.714		–		.714			–		.587	

(a) All features. (b) Dependency path features removed. (c) URL class removed.

Table 6: Binary and metaclassifier performance on all classes and all hand-labeled instances. Table (a) provides a benchmark for 10-class classification over highly substitutable query phrases. Table (b) shows that a lot of our performance can be achieved without computationally-expensive parsing.

	binary		meta		data
class	maxf	auc	prec	rec	%
no rel	.758	.719	.660	.882	57.8
synonym	.431	.901	.617	.199	2.4
hypernym	.284	.803	.367	.061	1.8
hyponym	.212	.804	.415	.056	1.6
coordinate	.588	.713	.615	.369	35.5
other	.206	.739	.375	.019	0.8
metaclassifier accuracy			.648		

Table 7: Binary and metaclassifier performance on WordNet-labeled instances with all features.

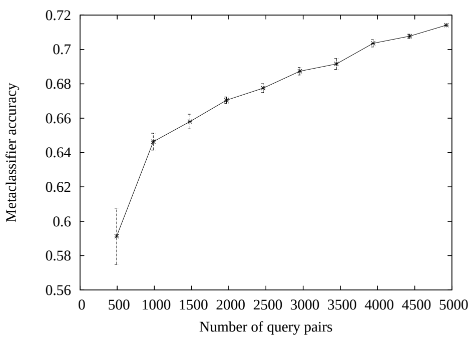

Figure 2: Meta-classifier accuracy as a function of number of labeled instances for training.

	binary		meta		data
class	maxf	auc	prec	rec	%
no rel	.525	.671	.485	.354	31.9
synonym	.381	.671	.684	.125	13.0
hypernym	.211	.605	0	0	6.2
hyponym	.125	.501	0	0	6.2
coordinate	.623	.628	.485	.844	42.6
metaclassifier accuracy			.490		

Table 8: Training on WordNet-labeled pairs and testing on hand-labeled pairs. Classifiers trained on WordNet do not generalize well.

6.4.2 Training on WordNet, Testing on WordNet and Hand-Labeled Pairs

We took the five classes for which human and WordNet definitions agreed (*synonyms, coordinates, hypernyms, hyponyms,* and *no relationship*) and trained classifiers on all WordNet-labeled instances. We tested the classifiers on human-labeled instances from just those five classes. Results are shown in Table 8. Performance was not very good, reinforcing the idea that while our features can distinguish between query segments, they cannot distinguish between common words.

7 Discussion

Almost all high-weighted features are either HTML paths or query log features; these are the ones that are easiest to obtain. Many of the highest-weight HTML tree features are symmetric, e.g. both words appear in cells of the same table, or as items in the same list. Here we note a selection of the more interesting predictors.

synonym —"X or Y" expressed as a dependency path was a high-weight feature.

hyper/hyponym —"Y and other X" as a dependency path has highest weight. An interesting feature is X in a table cell and Y appearing in text outside but nearby the table.

sibling —many symmetric HTML features. "X to the Y" as in "80s to the 90s". "X and Y", "X, Y, and Z" highly-weighted minipar paths.

general/specialization —the top three features are substring containment, word subset difference count, and prefix overlap.

spelling change —many negative features, indi-

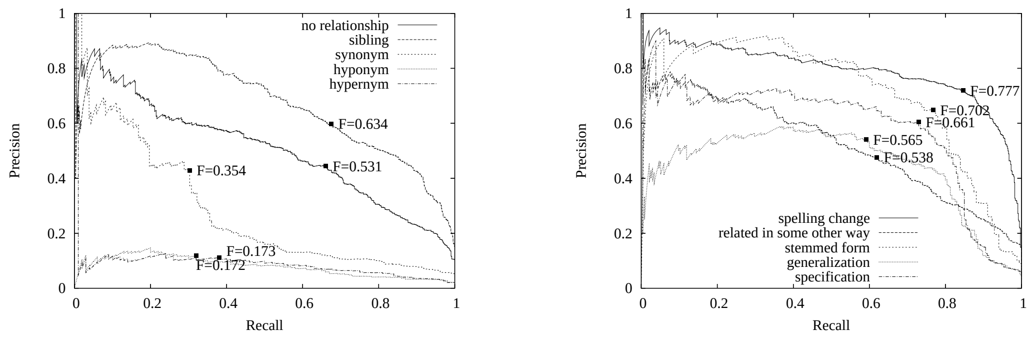

Figure 1: Precision-recall curves for 10 binary classifiers on all hand-labeled instances with all features.

cating that two words that cooccur in a web page are *not* likely to be spelling differences.

other —many symmetric HTML features. Two words emphasized in the same way (e.g. both bolded) may indicate some relationship.

none —many asymmetric HTML features, e.g. one word in a blockquote, the other bolded in a different paragraph. Dice coefficient is a good negative features.

8 Conclusion

We have provided the first benchmark for n-class semantic classification of highly substitutable query phrases. There is much room for improvement, and we expect that this baseline will be surpassed.

Acknowledgments

Thanks to Chris Manning and Omid Madani for helpful comments, to Omid Madani for providing the classification code, to Rion Snow for providing the hypernym data, and to our labelers.

This work was supported in part by the CIIR and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR001-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Peter G. Anick. 2003. Using terminological feedback for web search refinement: a log-based study. In *SIGIR 2003*, pages 88–95.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ted E. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of Coling 1992*, pages 539–545.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *15th International World Wide Web Conference (WWW-2006)*, Edinburgh.

Sathiya Keerthi and Dennis DeCoste. 2005. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, March.

Lillian Lee. 1999. Measures of distributional similarity. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

J. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. pages 61–74.

Soo Young Rieh and Hong Iris Xie. 2001. Patterns and sequences of multiple query reformulations in web searching: A preliminary study. In *Proceedings of the 64th Annual Meeting of the American Society for Information Science and Technology Vol. 38*, pages 246–255.

Rion Snow, Dan Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the Nineteenth Annual Conference on Neural Information Processing Systems (NIPS 2005)*.

Egidio Terra and Charles L. A. Clarke. 2004. Scoring missing terms in information retrieval tasks. In *CIKM 2004*, pages 50–58.

P.D Turney, M.L. Littman, J. Bigham, and V. Shnayder, 2003. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, chapter Combining independent modules in lexical multiple-choice problems, pages 101–110. John Benjamins.