Statistical resolution of scope ambiguity in natural language

Galen Andrew and Bill MacCartney

4 March 2004

1 Introduction

An unsolved problem in computational semantics is determining the relative scope precedence of multiple quantifiers and negations. Humans usually resolve these ambiguities effortlessly, but we have difficulty articulating the rules we use to do so. Scope ambiguities thus present a significant obstacle to automatically deriving formal semantic representations.

2 Data set generation

A crucial obstacle is that there is no labeled corpus available that is suitable for learning to resolve scope ambiguities. As a consequence, we've had to generate our own data set by (a) selecting sentences appropriate to our purposes, and (b) hand-labeling the sentences selected. We recognize that this represents a methodological compromise: our results would carry more weight if the training data, and especially the test data, had not been generated by the researchers themselves. However, we've attempted to mitigate this shortcoming by establishing in advance clear guidelines for selecting and labeling sentences.

We obtained our data from sentences drawn from GRE and LSAT logic games. We chose this limited domain because these sentences are specifically designed to have an interpretation that is unambiguous to a human reader. Thus there is less subjectivity about the correct labelings, and determining the correct scoping is less likely to depend on context and pragmatics.

2.1 Sentence selection

Another advantage of choosing LSAT logic games as our domain is that such problems use quantifiers frequently, and therefore provide a rich source of sentences containing multiple quantifiers.

We started with 122 LSAT logic games published during the 1980s and 1990s. These were scanned, OCRed, and edited to produce clean electronic copies. A Perl script then ran over the resulting files to extract sentences and to filter candidates for inclusion in the data set. The filter removed from consideration any sentences that did not contain at least two of a predefined set of designated quantifiers.

This list of designated quantifiers is presented in Table (1). While its exact composition is somewhat arbitrary, it includes all of the quantifying words and phrases most commonly used in the LSAT logic games. Three omissions deserve mention. First, we have not treated specific numbers as quantifiers, but only as parameters to quantifier groups such as 'at least'. Also, while some semanticists and philosophers of language, following Russell, treat *the* as a quantifier, we elected not to include it on the designated list, on the grounds that its frequency is too high and its usages too varied. Similarly, we have not considered proper names as quantifiers, although Montague semanticists would do so.

Finally, the list of candidate sentences was manually reviewed and edited, with the aim of removing inappropriate sentences and harvesting as much usable data as possible. Since our

quantifier	comment
a	*includes* an
all	
any	
at least	*includes* at least one, at least two, ...
at most	*includes* at most one, at most two, ...
both	
each	*excludes reciprocal usages such as* each other
either	
every	
exactly	*includes* exactly one, exactly two, ...
more than	*includes* more than one, more than two, ...
neither	
no	
none	
some	

Table 1: Quantifiers included in this study

goal was a data set in which each sentence contains exactly two designated quantifiers, much of the work in this phase focused on handling sentences containing three or more quantifiers. The following guidelines were followed:

- **Filtering.** We decided to remove sentences expressing questions, in part because LSAT questions employ certain standard formulas (e.g. 'which of the following is an acceptable list...') with high frequency, which might artificially skew results. An exception is that the antecedent of an 'if-then' question could sometimes be broken out as an independent two-quantifier sentence—see 'Splitting', next.

- **Splitting.** Many sentences containing four designated quantifiers are easily split into two independent clauses, each containing two designated quantifiers. To maximize the quantity of usable data, we effected such splits where possible. Example: *Each table has at least two sponsors seated at it, and each sponsor is seated at exactly one table.*

- **Cutting.** In many cases, a sentence with three or more quantifiers was easily converted to a two-quantifier sentence by cutting some part of the sentence—such as a dependent clause—without corrupting its meaning. For example, we were able to salvage the sentence "Each session will be taught by exactly one nurse and exactly one psychologist." by cutting the last four words.

- **Replacing.** Finally, many sentences with three or more quantifiers could be reduced to two-quantifier sentences by replacing some part of the sentence with a blank token. For example, the three-quantifier sentence "Any table at which both L and V are seated also has a third sponsor seated at it." can be reduced to a two-quantifier sentence by replacing "a third sponsor" by "ANOTHER". In these cases, care was taken to insure that the replacement word or phrase fulfilled the same syntactic role as the original.

2.2 Sentence labeling

The selected sentences were hand-labeled with one of four classes:

- **FIRST.** The first quantifier in the sentence takes wide scope. Example: "Each mannequin wears a hat." This is the most common case.

- **SECOND.** The second quantifier takes wide scope. Example: "Exactly six executives meet in a conference room."
- **EQUIVALENT.** Either quantifier may be read with wide scope, with a semantically equivalent result. Example: "A car drives into the center ring of a circus."
- **INDEPENDENT.** Although there are two quantifiers, their scopes are actually disjoint. Example: "K is on both committees and the planting committee has exactly three members."

The FIRST class comprised roughly 70% of the example sentences, while the other three classes comprised about 10% each.

2.3 Paucity and skew of data

Despite our strenuous efforts to salvage (in a principled fashion) as many example sentences as possible, we still face a shortage of data. From 122 LSAT logic puzzles, we were able to harvest 305 two-quantifier sentences. The strong skew in the class distribution—with over two-thirds of the examples labeled FIRST, and only one-tenth labeled SECOND—exacerbates the problem. However, we observe that such difficulties are characteristic of many important machine-learning applications and may be unavoidable.

3 Features

While automated statistical resolution of quantifier scope ambiguity has not been attempted (as far as we know), some research has been done in the linguistics and psycholinguistics communities regarding the way humans perform the task. Alshawi (1992) contains an extended discussion of *rules* that any resolution must follow, as well as softer *preferences* that are often followed, but may be overridden by other rules or preferences. Kurtzman and MacDonald (1993) present an empirical study into the question of which syntactic features humans seem to be using in a quantifier scope decision task. These two references were valuable in identifying features that are likely to be predictive.

In order to insure that all of our chosen machine learning algorithms (see sec. 4) could run on our data without modification, we chose to use real-valued features that did not employ an arbitrary ordering of non-ordinal values (e.g. the exact quantifying expression that occurs in a particular position). As is common in NLP applications, our features are all binary valued, zero or one depending on whether the instance exhibits a particular characteristic.

3.1 Occurrence of designated quantifiers

The features expected to be most strongly predictive are those which represent, for each of our designated quantifiying expressions, whether that expression appears in the sentence, in each position (first or second). With 15 quantifying expressions and two positions, there are a total of 30 of these features. The literature we examined suggests that certain quantifying words and phrases exhibit a strong preference for wide (or narrow) scope. For example, the quantifier *each* seems to prefer wide scope, (irrespective of position), while the quantifiers *at most*, *at least*, and *exactly* usually take narrow scope.

3.2 Syntactic/semantic features

We implemented several more features based on the syntactic parse of the instance sentence. First, the relation of each quantifying expression's enclosing noun phrase to the verb phrase on which it depends. In particular, we looked at whether the quantifier appears as the subject of

a verb and whether it appears as the object. Another feature was whether each quantifier is under the scope of a negation. Finally, we considered whether the two expressions appear in two separate phrases, joined by a conjunction, a case which we expected to be a strong indicator that the instance should be labeled "INDEPENDENT". All of these features are mentioned in the literature as having a strong influence on quantifier scope.

To extract these features, we used the most recent PCFG parser from Chris Manning's NLP group at Stanford, based on work by Klein and Manning (2003). Our impression was that the parser made very few, if any, errors on our data. We then used heuristics to read the feature values off of the parse tree using tgrep, a utility for identifying whether a parse tree exhibits a particular pattern. For example, a quantifier was considered to be the subject of a verb if it appeared as the head of a noun phrase with a verb phrase as an immediate right sister.

4 Learning algorithms

We evaluated and compared the performance of three different learning algorithms: Naive Bayes, logistic regression, and support vector machines (SVMs), all discriminative learning algorithms. While generative algorithms are sometimes appropriate for NLP problems such as parsing, in recent years the impression in NLP research is that discriminative algorithms yield better results for most applications.

4.1 Naive Bayes

We used a straightforward implementation of Naive Bayes (adapted from the Problem Set 2 spam classification task) with Laplace smoothing. As compared with the other learning algorithms, Naive Bayes gave excellent results (see section 6) and was very fast. We optimized the smoothing parameter using k-fold cross validation (with $k = 10$). Somewhat to our surprise, we discovered that a comparatively low value (about 0.1) gave the best results.

4.2 Logistic regression

Our initial implementation of logistic regression was also quite straightforward (adapted from the Problem Set 1 programming task). Initially, we tried using the Newton-Raphson method to maximize log likelihood as a function of the model parameters θ; however, we found that, while performing k-fold cross-validation during development, the Hessian was frequently singular and therefore non-invertible. This problem seemed to be a result of the sparsity of our feature representation and the paucity of data. Using gradient ascent instead of Newton-Raphson solved this problem, but ran frustratingly slowly. In the end, the solution was to add L_2 regularization to the logistic regression. We optimized the tradeoff parameter C using k-fold cross validation ($k = 10$), and found that a value of $C = 2$ worked best.

4.3 Support vector machines

Finally, we also tried using an SVM with L_2 regularization. As with logistic regression, the tradeoff parameter C was optimized using k-fold cross validation ($k = 10$). A value of $C = 2$ gave the lowest expected generalization error. Parameter fitting was performed using John Platt's SMO algorithm. One small hitch was that, for some folds of k-fold cross-validation, SMO failed to converge in any reasonable length of time. We were unable to find the source of this problem, but we were able to work around it, and it didn't seem to have much adverse affect on the results.

5 Experiment design

We attempted to learn two discriminative tasks, using cross-validation for development and a held-out test set for final test results.

5.1 Tasks

Because our data was labeled with four different classes, but we wanted to frame our problem as a binary classification task, we broke the challenge into two separate tasks:

- Task 1: Distinguish examples in which scoping precedence *matters* in terms of the correct semantic interpretation (classes FIRST and SECOND) from those in which it does not (classes EQUIVALENT and INDEPENDENT).
- Task 2: Among those examples where scoping precedence matters, distinguish those where the first quantifier should take wide scope (class FIRST) from those where the second quantifier should take wide scope (class SECOND).

Task 2 was our principal focus, and our feature engineering was directed primarily at the goal of performing well on this task.

5.2 Development methodology

During the development phase, we used k-fold cross-validation (within our training data) extensively as a guide for feature engineering and as a means of optimizing hyperparameters (e.g. smoothing and regularization parameters). Our standard value for k was 10, but we also tried other values (such as 5 and 20) for some runs. Our data set was small enough that it was also possible to try leave-one-out cross validation ($k = m$), although it was too slow to do on every run.

5.3 Testing methodology

For generating final test results, we used a held-out test set representing 15% of the original data set (46 examples). The examples in this test set were a contiguous randomly selected section of our entire data set. They were separated out immediately after the creation of the data set, and were kept "in the vault" throughout the development phase. The experimental results reported in section 6 are therefore based on training on the remaining 85% of the data (259 examples) and testing on this held-out test set.

6 Experimental results

6.1 Baseline

As a baseline for comparison, we used a method which simply applied the class label FIRST to every training example. (Thus, for task 1, the baseline mathod would predict that scoping matters, and for task 2, it would predict that the first quantifier takes wide scope.) The choice of this baseline metric is justified by the strong preference, in English, for the first quantifier in the sentence to take wide scope, and was strongly supported by the data. The baseline method made the correct prediction for more than 3/4 of examples on both tasks (see table 2).

Method	*Task 1*	*Task 2*
Baseline	76.1%	82.9%
Naive Bayes	65.2%	94.3%
Logistic regression	76.1%	91.4%
SVM	58.7%	94.3%

Table 2: Accuracy on held-out test set

7 Conclusions and future work

As is clear from table 2, our algorithms performed decently on task 2: We acheived a 2/3 reduction in error rate relative to the baseline. While there is room for improvement, we see this as an encouraging initial result.

Performance on task 1 however, was disappointing. We were unable to ourperform the baseline—indeed two of our three algorithms did not reach baseline accuracy. This may be a result of the fact that our features were designed largely with task 2 in mind. However, we observe that, with respect to the larger goal of automatically generating correct formal semantic representations, task 2 is by far the more important one. By definition, making an error in task 1 could have no impact on the quality of the semantic interpretation.

We see this project as a first step toward the goal of acheiving reliable automated quantifier scope resolution, and we intend to continue the line of research. A top priority will be to expand the quantity of data available for training and testing, allowing more accurate modeling of less frequent quantifying expressions. Also, we expect that our feature set could beneficially be expanded, for example to include the relationship of the quantifiers to modal expressions or passive constructions. We intend to perform a more detailed error analysis in order to identify opportunities for improvements in this area. As far as modeling, it has been suggested to us that maximum entropy models may facilitate modeling certain types of "X-OR" relationships between the quantifying expressions and scope ordering.

We hope that this research will eventually be a useful component in the larger task of automatic derivation of formal semantic representations.

8 Bibliography

Moran, D. & Pereira, F. (1992). Quantifier scoping. In Alshawi, Hiyan (Ed.). *The Core Language Engine* (pp. 149-172). MIT Press.

Kurtzman, H. & MacDonald, M. (1993). Resolution of quantifier scope ambiguities. *Cognition*, 48, 243-279.

Klein, D. & Manning, C. *Fast Exact Natural Language Parsing with a Factored Model*, Advances in Neural Information Processing Systems 15 (NIPS-2002), 2002.