Unrestricted Quantifier Scope Disambiguation

Mehdi Manshadi and James Allen
Department of Computer Science, University of Rochester
Rochester, NY, 14627, USA
{mehdih,james}@cs.rochester.edu

Abstract

We present the first work on applying statistical techniques to unrestricted Quantifier Scope Disambiguation (QSD), where there is no restriction on the type or the number of quantifiers in the sentence. We formulate unrestricted QSD as learning to build a Directed Acyclic Graph (DAG) and define evaluation metrics based on the properties of DAGs. Previous work on statistical scope disambiguation is very limited, only considering sentences with two explicitly quantified noun phrases (NPs). In addition, they only handle a restricted list of quantifiers. In our system, all NPs, explicitly quantified or not (e.g. definites, bare singulars/plurals, etc.), are considered for possible scope interactions. We present early results on applying a simple model to a small corpus. The preliminary results are encouraging, and we hope will motivate further research in this area.

1 Introduction

There are at least two interpretations for the following sentence:

(1) Every line ends with a digit.

In one reading, there is a unique digit (say 2) at the end of all lines. This is the case where the quantifier *A outscopes* (aka having *wide-scope* over) the quantifier *Every*. The other case is the one in which *Every* has *wide-scope* (or alternatively *A* has *narrow-scope*), and represents the reading in which different lines could possibly end with distinct digits. This phenomenon is known as *quantifier scope ambiguity*.

Shortly after the first efforts to build natural language understanding systems, Quantifier Scope Disambiguation (QSD) was realized to be very difficult. Woods (1978) was one of the first to suggest a way to get around this problem. He presented a framework for scope-underspecified semantic representation. He suggests representing the Logical Form (LF) of the above sentence as:

(2) *<Every x Line>*
<A y Digit>
Ends-with(x, y)

in which, the relative scope of the quantifiers is underspecified. Since then *scope underspecification* has been the most popular way to deal with quantifier scope ambiguity in deep language understanding systems (e.g. Boxer (Bos 2004), TRAINS (Allen et al. 2007), BLUE (Clark and Harrison 2008), and DELPH-IN[1]). Scope underspecification works in practice, only because many NLP applications (e.g. machine translation) could be achieved without quantifier scope disambiguation. QSD on the other hand, is critical for many other NLP tasks such as question answering systems, dialogue systems and computing entailment.

Almost all efforts in the 80s and 90s on QSD adopt heuristics based on the lexical properties of the quantifiers, syntactic/semantic properties of the sentences, and discourse/pragmatic cues (VanLehn

[1] http://www.delph-in.net/

Proceedings of the TextGraphs-6 Workshop, pages 51–59,
Portland, Oregon, USA, 19-24 June 2011. ©2011 Association for Computational Linguistics

1978, Moran 1988, Alshawi 1992). For example, it is widely known that in English, the quantifier *each* tends to have the widest scope. Also, the subject of a sentence often outscopes the direct object.[2] In cases where these heuristics conflict, (manually) weighted preference rules are adopted to resolve the conflict (Hurum 1988, , Pafel 1997).

In the last decade there has been some effort to apply statistical and machine learning (ML) techniques to QSD. All the previous efforts, however, suffer from the following two limitations (see section 2 for details):

- They only allow scoping two NPs per sentence.
- The NPs must be explicitly quantified (e.g. they ignore definites or bare singulars/plurals), and the quantifiers are restricted to a predefined list.

In this paper, we present the first work on applying statistical techniques to unrestricted QSD, where we put no restriction on the type or the number of NPs to be scoped in a sentence. In fact, every two NPs, explicitly quantified or not (including definites, indefinites, bare singulars/plurals, pronouns, etc.), are examined for possible scope interactions. Scoping only two quantifiers per sentence, the previous work defines QSD as a single classification task (e.g. 0 where the first quantifier has wide-scope, and 1 otherwise). As a result standard metrics for classification tasks are used for evaluation purposes. We formalize the unrestricted form of QSD as learning to build a DAG over the set of NP chunks in the sentence. We define accuracy, precision and recall metrics based on the properties of DAGs for evaluation purposes.

We report the application of our model to a small corpus. As seen later, the early results are promising and shall motivate further research on applying ML techniques to unrestricted QSD. In fact, they set a baseline for future work in this area.

The structure of this paper is as follows. Section (2) reviews the related work. In (3) we briefly describe our corpus. We formalize the problem of quantifier scope disambiguation for multiple quantifiers in section (4) and define some evaluation metrics in (5). (6) presents our model including the kinds of features we have used. We present our experiments in (7) and give a discussion of the results in (8). (9) summarizes the current work and gives some directions for the future work.

[2] Allen (1995) discusses some of these heuristics and gives an algorithm to incorporate those for scoping while parsing.

2 Related work

Earlier we mentioned that a standard approach to deal with quantifier scope ambiguity is scope underspecification. More recent underspecification formalisms such as Hole Semantics (Bos 1996), Minimal Recursion Semantics (Copestake et al. 2001), and Dominance Constraints (Egg et al. 2001), present constraint-based frameworks. Every constraint forces one term to be in the scope of another, hence filters out some of the possible readings. For example, one may add a constraint to an underspecified representation (UR) to force island constraints. Constraints can be added incrementally to the UR as the sentence processing goes deeper (e.g. at the discourse and/or pragmatic level). The main drawback with these formalisms is that they only allow for hard constraints; that is every scope-resolved representation must satisfy all the constraints in order to be a valid interpretation of the sentence. In practice, however, most constraints that can be drawn from discourse or pragmatic knowledge have a soft nature; that is, they describe a scope preference that is allowed to be violated, though at a cost.

Motivated by the above problem, Koller et al. (2008) define an underspecified scope representation based on regular tree grammars, which allows for both hard constraints and weighted soft constraints. They present a PCFG-style algorithm that computes the reading, which satisfies all the hard constraints and has the maximum product of the weights. However, they assume that the weights are already given. Their algorithm, for example, can be used in traditional QSD approaches with weighted heuristics to systematically compute the best reading. The main question though is how to automatically learn those weights. One solution is using corpus-based methods to learn soft constraints and the cost associated with their violation, in terms of *features* and their *weights*.

To the best of our knowledge, there have been three major efforts on statistical scope disambiguation for English. Higgins and Sadock (2003), hence HS03, is the first work among these systems. They define a list of quantifiers that they consider for scope disambiguation. This list does not include definites, indefinites, and many other challenging scope phenomena. They extract all sentences from the Wall Street journal section of the Penn Treebank, containing exactly two quantifiers from this

list. This forms a corpus of 890 sentences, each labeled with the relative scope of the two quantifiers, with the possibility of no scope interaction. The no scope interaction case happens to be the majority class in their corpus and includes more than 61% of the sentences, defining a baseline for their QSD system. They achieve the inter-annotator agreement of only 52% on this task.

They treat QSD as a classification task with three possible classes (*wide scope*, *narrow scope*, and *no scope interaction*). Three forms of feature are incorporated into the classifier: part-of-speech (POS) tags, lexical features, and syntactic properties. Several classification models including naïve Bayes classifier, maximum entropy classifier, and single-layer perceptron are tested, among which the single-layer perceptron performs the best, with the accuracy of 77%.

Galen and MacCartney (2004), hence GM04, build a corpus of 305 sentences from LSAT and GRE logic games, each containing exactly two quantifiers from an even more restricted list of quantifiers. They use an additional label for the case where the two scopings are equivalent (as in the case of two existentials). In around 70% of the sentences in their corpus, the first quantifier has wide scope, defining a majority class baseline of 70% for their QSD system.[3] Three classifiers are tried: naïve Bayes, logistics regression, and support vector machine (SVM), among which SVM performs the best and achieves the accuracy of 94%.

In a recent work, Srinivasan and Yates (2009) study the usage of pragmatic knowledge in finding the preferred scoping of natural language sentences. The sentences are all extracted from 5-grams in Web1Tgram (from Google, Inc) and share the same syntactic structure: an active voice English sentence of the form *(S (NP (V (NP | PP))))*. For the task of finding the most preferred reading, they annotate 46 sentences, each containing two quantifiers: *Every* and *A*, where the first quantifier is always *A*. Each sentence is annotated with one of the two labels (*Every* has wide scope or not). They use a totally different approach for finding the preferred reading. The n-grams in Web1Tgram are used to extract relations such as *Live(Person, City)*, and to estimate the expected cardinality of the two classes, which form the arguments of the relation, that is *Person* and *City*.

[3] They do not report any inter-annotator agreement.

1. *Print [1/ every line] of [2/ the file] that starts with [3/ a digit] followed by [4/ punctuation].*
   QSD: {2>1, 2>3, 1>3, 2>4, 1>4}
2. *Delete [1/ the first character] of [2/ every word] and [3/ the first word] of [4/ every line] in [5/ the file].*
   QSD: {5>4, 5>3, 4>3, 5>2, 5>1, 2>1}

Figure 1. Two NP-chunked sentences with QSDs

They decide on the preferred scoping by comparing the size of the two classes, achieving the accuracy of 74% on their test set. The main advantage of this work is that it is open domain.

3 Our corpus

The fact that HS03, in spite of ignoring challenging scope phenomena and scoping only two quantifiers per sentence, achieve the IAA of 52% shows how hard scope disambiguation could be for humans. It becomes enormously more challenging when there is no restriction on the type or the number of quantifiers in the sentence, especially when NPs without explicit quantifiers such as definites, indefinites, and bare singulars/plurals are taken into account. As a matter of fact, our own early effort to annotate part of the Penn Treebank with full scope information soon proved to be too ambitious. Instead, we picked a domain that covers most challenging phenomena in scope disambiguation, while keeping the scope disambiguation fairly intuitive. This made building the first corpus of English text with full quantifier scope information feasible. Our domain of choice is the description of tasks about editing plain text files, in other words, a natural language interface for text editors such as SED, AWK, or EMACS. Figure (1) gives some sentences from the corpus. The reason behind scoping in this domain being fairly intuitive is that given any of these sentences, a conscious knowledge of scoping is critical in order to be able to accomplish the explained task.

Our corpus consists of 500 sentences manually extracted from the web. The sentences have been labeled with gold standard NP chunks, where each NP chunk has been indexed with a number 1 through n (n is the number of chunks in the sentence). The annotators are asked to use outscoping relations represented by '>' to specify the relative scope of every pair $1 \le i,j \le n$, with an option to leave

the pair *unscoped*. For example a relation (*2>3*) states that the second NP in the sentence *outscopes* (aka *dominates*) the third NP. Since outscoping relation is *transitive*, for the convenience of the annotation, the outscoping relations are allowed to be cascaded forming dominance chains. For example, the scoping for the sentence 2 in figure (1) can alternatively be represented as shown in (3).

(3) (5>2>1 ; 5>4>3)

As a result, every pair $<i,j>$ $(1 \le i < j \le n)$ is implicitly labeled with one of the three labels:

i. *Wide scope:* either explicitly given by the annotator as *i>j* or implied using the transitive property of outscoping[4]

ii. *Narrow scope:* either explicitly given by the annotator as *j>i* or implied using the transitive property of outscoping

iii. *No interaction:* where neither wide scope nor narrow scope could be inferred from the given scoping.[5]

We achieved the IAA of 75% (based on Cohen's kappa score) on this corpus, significantly better than the 52% IAA of HS03, especially considering the fact that we put no restriction on the type of the quantification. Our sentence-level IAA is around 66%. The details of the corpus, and the annotation scheme are beyond the scope of this paper and can be found in Manshadi et al. (2011).

4 Formalization

Outscoping is an *anti-symmetric transitive* relation, so it defines an *order* over the chunks. Since we do not force every two chunks to be involved in an outscoping relation, QSD defines a *partial order* over the NP chunks. Formally,

Definition 1: Given a sentence S with NP chunks *1..n*, a relation *P* over *{1..n}* is called a *QSD* for S, if and only if *P* is a partial order.

Definition 2: Given a sentence S with NP chunks *1..n*, and the QSD *P*, we say (chunk) *i outscopes* (chunk) *j* if and only if $(i>j) \in P$.

[4] That is if *i* outscopes *j* and *j* outscopes *k* then *i* outscopes *k*.

[5] The no interaction class includes two cases: no scope interaction and logical equivalence which means we follow the three-label scheme of HS03 as opposed to the four-label scheme of GM04. This is because when there is a logical equivalence, except for trivial cases, there are no clear criteria based on which one can decide whether there is a scope interaction or not. Furthermore, distinguishing these two cases does not make much difference in practice.

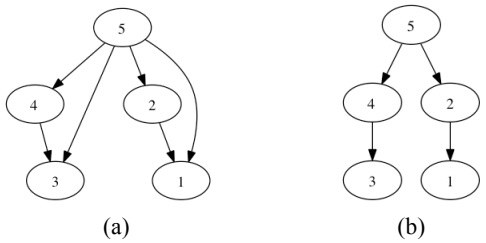

Figure 2. Scopings represented as DAGs

Definition 3: Given a sentence S with NP chunks *1..n*, and the QSD *P*, chunk *i* is said to be *disjoint* with chunk *j* if and only if

$(i>j) \notin P \wedge (j>i) \notin P$.

4.1 QSD and directed acyclic graphs

Partial orders can be represented using *Directed Acyclic Graphs* (*DAG*s) in which *dominance* (aka *reachability*) determines the order. More precisely, every DAG *G* over *n* nodes $v_1..v_n$ defines a partial order P_G over the set $\{v_1..v_n\}$ in which, v_i precedes v_j in P_G if and only if v_i *dominates*[6] v_j in *G*.

Definition 4: Given a sentence S with NP chunks *1..n*, every DAG *G* over *n* nodes (labeled *1...n*) defines a QSD P_G for S, such that

$(i>j) \in P_G \Leftrightarrow$ *i dominates j in G*

For example figure (2a,b) represent the DAGs corresponding to the QSD of sentence 2 in figure (1) and the QSD in (3) respectively. Following definition 3 and 4, the no interaction relation defined in section (3) translates to corresponding nodes in the DAG being *disjoint*[7]. Therefore the three types of scope interaction defined in *i*, *ii*, and *iii* (section 3), translate to the following relations in a DAG.

(4) Wide Scope (WS): i dominates j
Narrow Scope (NS): j dominates i
No Interaction (NI): i and j are disjoint.

5 Evaluation metrics

Intuitively the similarity of two QSDs, given for a sentence S, can be defined as the ratio of the chunk pairs that have the same label in both QSDs to the total number of pairs. For example, consider the

[6] Given a DAG *G=(V, E)*, node *u* is said to immediately dominate node *v* if and only if $(u,v) \in E$. "dominates" is the *reflexive transitive* closure of "immediately dominates".

[7] The nodes *u* and *v* of the DAG *G* are said to be disjoint if neither *u* dominates *v* nor *v* dominates *u*.

two DAGs in figure (2). Although looking different, both DAGs define the same partial order (i.e. QSD). This is because the partial order represented by a DAG G corresponds to the *transitive closure* (*TC*) of G.

5.1 Transitive closure

The transitive closure of G, shown as G^+, is defined as follows:

(5) $G^+ = \{(i,j) \mid i \text{ dominates } j \text{ in } G\}$

For example, figure (2a) is the transitive closure of the DAG in figure (2b). Given this, the similarity metric mentioned above can be formally defined as the number of (unordered) pairs of node that match between G_1^+ and G_2^+ divided by the total number of (unordered) pairs.

Definition 5: Similarity measure or σ.
Given sentence S with n NP chunks and two scopings represented by DAGs G_1 and G_2, we define:

$$M(G_1, G_2) = \{ \{i,j\} \mid ((i,j) \in G_1^+ \wedge (i,j) \in G_2^+) \vee ((j,i) \in G_1^+ \wedge (j,i) \in G_2^+) \vee ((i,j),(j,i) \notin G_1^+ \wedge (i,j),(j,i) \notin G_2^+) \}$$

$$\sigma(G_1, G_2) = 2|M(G_1, G_2)| / [n(n-1)]$$

Where $|.|$ represents the cardinality of a set. σ is a value between 0 and 1 (inclusive) where 1 means that the QSDs are equivalent and 0 means that they do not agree on the label of any pair. σ is useful for measuring the similarity of two scope annotations when calculating IAA. It can also be used as an *accuracy* metric for evaluating an automatic scope disambiguation system where the similarity of a predicted QSD is calculated respect to a gold standard QSD. In fact, if $n = 2$, σ is equivalent to the metric that HS03 use to evaluate their system.

The similarity metric defined above has some disadvantages. For example, HS03 report that more than 61% of the scope relations in their corpus are of type no interaction. Using this metric, a model that leaves everything unscoped has more than 61% percent accuracy on their corpus! In fact, the output of a QSD system on pairs with no interaction is not practically important.[8] What is more important is to recover the pairs with scope interaction correctly. The standard way to address this is to define precision/recall metrics.

Definition 6: Precision and Recall (TC version)
Given the gold standard DAG G_g and the predicted DAG G_p, we define the precision (P) and the recall (R) as follows:

$$TP = |\{ (i,j) \mid (i,j) \in G_p^+ \wedge (i,j) \in G_g^+ \}|$$
$$N = |\{ (i,j) \mid (i,j) \in G_p^+ \}|$$
$$M = |\{ (i,j) \mid (i,j) \in G_g^+ \}|$$
$$P = TP / N$$
$$R = TP / M$$

5.2 Transitive reduction

The TC-based metrics implicitly count some matching pairs more than once. For example, if in both QSDs we have 1>2 and 2>3, then 1>3 is implied, so counting it as another match is redundant and favors toward higher accuracies. Naturally, there are so many redundancies in TC. To address this issue, we define another set of metrics based on the concept of *transitive reduction* (TR). Given a directed graph G, the transitive reduction of G, represented as G^-, is intuitively a graph with the same reachability (i.e. dominance) relation but with no redundant edges. More formally, the transitive reduction of G is a graph G^- such that

- $(G^-)^+ = G^+$
- $\forall G', \quad (G')^+ = G^+ \Rightarrow |G^-| \leq |G'|$

For example, figure (2b) represents the transitive reduction of the DAG in figure (2a). Fortunately if a directed graph is acyclic, its transitive reduction is unique (Aho et al., 1972). Therefore, defining TR-based precision/recall metrics is valid.

Definition 7: Precision and Recall (TR version)
Simply replace every '+' in definition 6 with a '-'.

6 The model

We extend HS03's approach for scoping two NPs per sentence to the general case of n NPs. Every pair of chunks $<i,j>$ (where $1 \leq i < j \leq n$) is treated as an independent sample to be classified as one of the three classes defined in (3), that is *WS*, *NS*, or *NI*. Therefore a sentence with n NP chunks consists of $C(n, 2) = n(n-1)/2$ samples. The average

[8] In practice the target language is often first order logic or a variant of that. When a pair is labeled *NI* in gold standard data, if there exist valid interpretations (satisfying hard constraints) in which either of the two quantifiers can be in the scope of the other, then the ordering of this pair does not matter; that is switching the order of such pairs result in equivalent formulas.

1. *Print [1/ every/D line/H] of [2/ the/D file/H] that starts with [3/ two/CD digits/H/S] followed by [4/ punctuation/H].*

2. *Delete [1/ the/D first character/H] of [2/ every/D word/H] and [3/ the/D first word/H] of [4/ every/D line/H] in [5/ the/D file/H].*

Figure 3. Labeling determiners and head nouns

number of NPs per sentence in the corpus is 3.7, so the corpus provides 1850 samples. Since the scoping of each pair is predicted independent of the other pairs in the sentence, it may result in an ill-formed scoping, i.e. a scoping with cycles. As explained later, this case did not happen in our corpus. A *MultiClass SVM* (Crammer et al. 2001), referred to as *SVM-MC* in the rest of the paper, is used as the classifier. We provide more supervision by annotating data with the following labels.

I. Determiner features

For every NP chunk, we tag pre-determiner (/PD), determiner (/D), possessive determiner (/POS), and number (/CD) (if they exist) as part of the determiner (see figure 3). Given the pair <*i,j*>, for either of the chunks *i* and *j,* and every tag mentioned above, we use a binary feature, which shows whether this tag exists in that chunk or not. For tags that do exist (except /CD) the lexical word is also used as a feature.

II. Semantic head features

We tag the semantic head of the NP and use its lexical word as feature. Also the plurality of the NP (/S tag for plurals) is used as a binary feature.

III. 3. Dependency features

The above two sets of feature are about the individual properties of the chunks. But this last category represents how each NP contributes to the semantics of the whole sentence. We borrow from Manshadi et al. (2009) the concept of *Dependency Graph* (*DG*), which encodes this information in a compact way. DG represents the argument structure of the predicates that form the logical form of a sentence. The DG of a sentence with *n* NP chunks contains *n+1* nodes labeled *0..n.* Node *i* (*i>0)* represents the predicate or the conjunction of the predicates that describes the NP chunk *i,* and node 0 represents the main predicate (or conjunction of predicates) of the sentence. An edge from *i* to *j* shows that chunk *i* is an argument of a predicate represented by node *j*.

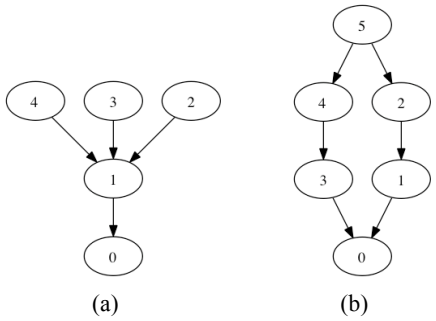

Figure 4. Dependency Graphs for figure (3) sentences

For example, in sentence (1) of figure (3), chunk 1 is clearly the argument of the verb *Print* (the main predicate of the sentence), therefore there is an edge from 1 to 0 in the DG of this sentence as shown in figure (4a). Also, chunks 2..4 are part of the description of chunk 1, so they are the arguments of the predicate(s) describing chunk 1. This means that there must be edges from nodes 2..4 to node 1 in the DG. Similarly for sentence 2 in figure (3), chunk 5 is part of the description (hence an argument of the predicates) of chunks 2 and 4; chunks 2 and 4 are part of the description of 1 and 3 respectively; and 1 and 3 are both arguments of the verb *Delete*, the main predicate of the sentence, resulting in the DG given in figure (4b).

The following features are extracted from the DG for every sample <*i,j*>(*1≤i<j≤n*):

- *Does i (or j) immediately dominate 0?*
- *Does i (or j) immediately dominate j (or i)?*
- *Does i (or j) dominate j (or i)?*
- *Are i,j siblings ?*
- *Do i,j share the same child?*

Note that DG has a close relationship with the dependency tree of a sentence; for example, it shows the dependency relation(s) between a noun or verb and their modifier(s). Therefore it actually encodes some syntactic properties of a sentence.

7 Experiments

100 sentences from the corpus were picked at random as the development set, in order to study the relevant features and their contribution to QSD. The rest of the corpus (400 sentences) was then used to do a 5 fold cross validation. We used $SVM^{Multiclass}$ from *SVM-light* toolkit (Joachims 1999) as the classifier.

	P	R	F
Baseline (TC)	31.8%	49.7%	38.8%
Baseline (TR)	27.4%	33.9%	30.3%
SVM-MC (TC)	73.0%	**84.7%**	78.4%
SVM-MC (TR)	70.6%	**76.2%**	73.2%

Table 1. Constraint-level results

Before giving the results, we define a baseline. HS03 use the most frequent label as the baseline and the similarity metric given in definition (5) to evaluate the performance. Since more than 61% of the labels in their corpus is *NI*, the baseline system (that leaves every sentence unscoped) has the accuracy above 61%. In our corpus, the majority class is *WS* containing around 35% of the samples. *NS* and *NI* each contain 34% and 31% of the samples respectively. This means that there is a slight tendency for having scope preference in chronological order. Therefore, the linear order of the chunks (i.e. from left to right) defines a reasonable baseline.

The results of our experiments are shown in table 1. The table lists the parameters *P*, *R*, and *F*-score[9] for our SVM-MC model vs. the baseline system. For each system, two sets of metrics have been reported: TC-based and TR-based.

Table 2 lists the sentence-level accuracy of the system. We computed two metrics for sentence-level accuracy: *Acc* and *Acc-EZ*. In calculating *Acc*, a sentence is considered correct if all the labels (including *NI*) exactly match the gold standard labels. However, this is an unnecessarily tough metric. As mentioned before (footnote 8), in practice the output of the system for the samples labeled *NI* is not important; all we care is that all *outscoping* (i.e. *WS/NS*) relations are recovered correctly. In other words, in practice, the system's recall is the most important parameter. Regarding this fact, we define *Acc-EZ* as the percentage of sentences with 100% recall (ignoring the value of precision).

In order to compare our system with that of HS03, we applied our model unmodified to their corpus using the same set-up, a 10-fold cross validation. However, since their corpus is not annotated with DG, we translated our dependency features to the properties of the Penn Treebank's phrase structure trees. Table (3) lists the accuracy

[9] F-score is defined as *F=2PR/(P+R)*.

	Acc	*Acc-EZ*
Baseline	27.0%	43.8%
SVM-MC	62.3%	78.0%

Table 2. Sentence level accuracy

of their best model, their baseline, and our SVM-MC model. As seen in this table, their model outperforms ours. This, however, is not surprising. First, although we trained our model on their corpus, the feature engineering of our model was done based on our own development set. Second, since our corpus is not annotated with phrase structure trees, our model does not use any of their features that can only be extracted from phrase structure trees. It remains for future work to incorporate the features extracted from phrase structure trees (which is not already encoded in DG) and evaluate the performance of the model on either corpus.

8 Discussion

As seen in tables 1 and 2, for a first effort at full quantifier scope disambiguation, the results are promising. The constraint-based F-score of 78% is already higher than the inter-annotator agreement, which is 75% (measured using the TC-based similarity metric; see definition 5). Furthermore, our system outperforms the baseline, by more than 40% (judging by the constraint-based F-score). This is significant, comparing to the work of HS03, which outperforms the baseline by 16%.

We mentioned before that in our corpus in average there are around 4 NPs per sentence resulting in 6 samples per sentence. Therefore the chance of predicting all the labels correctly is very slim. However, the baseline (i.e. the left to right order) does a good job and predicts the correct QSD for 27% of the sentences. At the sentence level, our model does not reach the IAA, but the performance (62%) is not much lower than the IAA (66%).

A question may arise that since the model treats

	σ
Baseline	61.1%
HS04	**77.0%**
Our Model	73.3%

Table 3. Comparison with HS04 system on their dataset

the pairs of NP independently, what guarantees that the scopings are valid; that is the predicted directed graphs are in fact DAGs. For example, for a sentence with 3 NP chunks, the classifier may predict that 1>2, 2>3, and 3>1, which results in a loop! As a matter of fact, there is nothing in the model that guarantees the validness of the predicted scopings. In spite of that, surprisingly all generated graphs in our tests were in fact DAGs! In order to explain this fact, we run two experiments. In the first experiment, corresponding to every sentence S in the corpus with *n* chunks, we generated a random directed graph over *n* nodes. Only 10% of the graphs had cycles. It means that more than 90% of randomly generated directed graphs with *n* nodes (where the distribution of *n* is its distribution in our corpus) are acyclic. In the second experiment, for every sentence with *n* chunks, we created the samples $<i,j>$ by randomly selecting values for all the features. We then tested the classifier in our original set-up, a 5-fold cross validation. In this case, only 4% of the sentences were assigned inconsistent labeling. This means that chances of having a loop in the scoping are small even when the classifier is trained on samples with randomly valued features, therefore it is not surprising that a classifier trained on the actual data learns some useful structures which make the chance of assigning inconsistent labels very slim.

In general, if the classifier predicts such inconsistent scopings, the PCFG-style algorithm of Koller et al. (2008) comes handy in order to find a valid scoping with the highest weight.

9 Summary and future work

We presented the first work on unrestricted statistical scope disambiguation in which all NP chunks in a sentence are considered for possible scope interactions. We defined the task of full scope disambiguation as assigning a directed acyclic graph over *n* nodes to a sentence with *n* NP chunks. We then defined some metrics for evaluation purposes based on the two well-known concepts for DAGs: transitive closure and transitive reduction.

We use a simple model for automatic QSD. Our model treats QSD as a ternary classification task on every pair of NP chunks. A multiclass SVM together with some POS, lexical and dependency features is used to do the classification. We apply this model to a corpus of English text in the domain of editing plain text files, which has been annotated with full scope information. The preliminary results reach the F-score of 73% (based on transitive reduction metrics) at the constraint level and the accuracy of 62% at the sentence level. The system outperforms the baseline by a high margin (43% at the constraint level and 35% at the sentence level).

Our ternary SVM-based classification model is a preliminary model, used for justification of our theoretical framework. Many improvements are possible, for example, directly predicting the whole DAG as a structured output. Also, the features that we use are rather basic. There are other linguistically motivated features that can be incorporated, e.g. some properties of the phrase structure trees, not already encoded in dependency graphs.

Another problem with the current system is that the extra supervision has been provided by manually labeling the data (e.g. with dependency graphs). This could be done automatically by applying off the shelf parsers or POS taggers, possibly by adapting them to our domain.

Although we consider all NPs for scope resolution, scopal operators such as *negation*, *modal/logical* operators have been ignored in this work. We also do not distinguish *distributive* vs. *collective* reading of plurals in the current system.[10] Incorporating scopal operators and handling distributivity vs. collectivity would be the next step in expanding this work.

Finally, since hand annotation of scope information is very challenging, applying semi-supervised or even unsupervised techniques to QSD is very demanding. In fact, leveraging unlabeled data to do QSD seems quite promising. This is because domain dependent knowledge plays a critical role in scope disambiguation and this knowledge can be learned from unlabeled data using unsupervised methods.

Acknowledgement

We would like to thank Derrick Higgins for providing us with the HS03's corpus. This work was supported in part by grants from the National Science Foundation (IIS-1012205) and The Office of Naval Research (N000141110417).

[10] The corpus has already been annotated with all this information, but our QSD model is not designed for such a comprehensive scope disambiguation.

References

Aho, A., Garey, M., Ullman, J. (1972). *The Transitive Reduction of a Directed Graph*. SIAM Journal on Computing 1 (2): 131–137.

Allen, J. (1995) *Natural Langue Understanding*, Benjamin-Cummings Publishing Co., Inc.

Allen, J., Dzikovska, M., Manshadi, M., Swift, M. (2007) *Deep linguistic processing for spoken dialogue systems*. Proceedings of the ACL-07 Workshop on Deep Linguistic Processing, pp. 49-56.

Alshawi, H. (ed.) (1992) *The core language Engine*. Cambridge, MA, MIT Press.

Bos, J., S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier (2004). *Wide-coverage semantic representations from a CCG parser*. In Proceedings of COLING 2004, Geneva, Switzerland, pp. 1240–1246.

Bos, J. (1996) *Predicate logic unplugged*. In Proc. 10th Amsterdam Colloquium, pages 133–143.

Clark P., Harrison, P. (2008) *Boeing's NLP system and the challenges of semantic representation*, Semantics in Text Processing. STEP 2008.

Copestake, A., Lascarides, A. and Flickinger, D. (2001) *An Algebra for Semantic Construction in Constraint-Based Grammars*. ACL-01. Toulouse, France.

Crammer, K., Y. Singer, N. Cristianini , J. Shawe-taylor, B. Williamson (2001). *On the Algorithmic Implementation of Multi-class SVMs*, Journal of Machine Learning Research.

Egg M., Koller A., and Niehren J. (2001) *The constraint language for lambda structures*. Journal of Logic, Language, and Information, 10:457–485.

Galen, A. and MacCartney, B. (2004). Statistical resolution of scope ambiguity in Natural language. http://nlp.stanford.edu/nlkr/scoper.pdf.

Higgins, D. and Sadock, J. (2003). *A machine learning approach to modeling scope preferences*. Computational Linguistics, 29(1).

Hurum, S. O. (1988) *Handling scope ambiguities in English.* In Proceeding of the second conference on Applied Natural Language Processing (ANLC '88).

Koller, A., Michaela, R., Thater, S. (2008) *Regular Tree Grammars as a Formalism for Scope Underspecification*. ACL-08, Columbus, USA.

Joachims, T. (1999) *Making Large-Scale SVM Learning Practical*. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press.

Manshadi, M., Allen J., and Swift, M. (2009) *An Efficient Enumeration Algorithm for Canonical Form Underspecified Semantic Representations*. Proceedings of the 14th Conference on Formal Grammar (FG 2009), Bordeaux, France July 25-26.

Moran, D. B. (1988). *Quantifier scoping in the SRI core language engine*. In Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics.

Pafel, J. (1997). *Skopus und logische Struktur. Studien zum Quantorenskopus im Deutschen*. PHD thesis, University of Tübingen.

Srinivasan, P., and Yates, A. (2009). *Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

VanLehn, K. (1988) *Determining the scope of English quantifiers*, TR AI-TR-483, AI Lab, MIT.

Woods, W. A. (1978) *Semantics and quantification in natural language question answering*, Advances in. Computers, vol. 17, pp 1-87.