Plurality, Negation, and Quantification: Towards Comprehensive Quantifier Scope Disambiguation

Mehdi Manshadi, Daniel Gildea, and James Allen
University of Rochester
734 Computer Studies Building
Rochester, NY 14627
mehdih,gildea,james@cs.rochester.edu

Abstract

Recent work on statistical quantifier scope disambiguation (QSD) has improved upon earlier work by scoping an arbitrary number and type of noun phrases. No corpus-based method, however, has yet addressed QSD when incorporating the implicit universal of plurals and/or operators such as negation. In this paper we report early, though promising, results for automatic QSD when handling both phenomena. We also present a general model for learning to build partial orders from a set of pairwise preferences. We give an $n \log n$ algorithm for finding a guaranteed approximation of the optimal solution, which works very well in practice. Finally, we significantly improve the performance of the previous model using a rich set of automatically generated features.

1 Introduction

The sentence *there is one faculty member in every graduate committee* is ambiguous with respect to quantifier scoping, since there are at least two possible readings: If *one* has **wide scope**, there is a unique faculty member on every committee. If *every* has wide scope, there can be different faculty members on each committee. Over the past decade there has been some work on statistical quantifier scope disambiguation (QSD) (Higgins and Sadock, 2003; Galen and MacCartney, 2004; Manshadi and Allen, 2011a). However, the extent of the work has been quite limited for several reasons. First, in the past two decades, the main focus of the NLP community has been on shallow text processing. As a deep processing task, QSD is not essential for many NLP applications that do not require deep understanding. Second, there has been a lack of comprehensive scope-disambiguated corpora, resulting in the lack of work on extensive statistical QSD. Third, QSD has often been considered only in the context of explicit quantification such as *each* and *every* versus *some* and *a/an*. These co-occurrences do not happen very often in real-life data. For example, Higgins and Sadock (2003) find fewer than 1000 sentences with two or more explicit quantifiers in the Wall Street journal section of Penn Treebank. Furthermore, for more than 60% of those sentences, the order of the quantifiers does not matter, either as a result of the logical equivalence (as in two existentials), or because they do not have any scope interaction.

Having said that, with deep language processing receiving more attention in recent years, QSD is becoming a real-life issue.[1] At the same time, new scope-disambiguated corpora have become available (Manshadi et al., 2011b). In this paper, we aim at tackling the third issue mentioned above. We push statistical QSD beyond explicit quantification, and address an interesting, yet practically important, problem in QSD: plurality and quantification. In spite of an extensive literature in theoretical semantics (Hamm and Hinrichs, 2010; Landmann, 2000), this topic has not been well investigated in computational linguistics. To illustrate the phenomenon, consider (1):

1. *Three words start with a capital letter.*

A deep understanding of this sentence, requires deciding whether each *word* in the set, referred to by *Three words*, starts with a potentially distinct *capital letter* (as in *Apple, Orange, Banana*) or there is a unique *capital letter* which each *word* starts with (as in *Apple, Adam, Athens*). By treating the NP *Three words* as a single atomic entity, earlier work on automatic QSD has overlooked this problem. In general, every plural NP potentially introduces an **implicit universal**, ranging

[1] For example, Liang et al. (2011) in their state-of-the-art statistical semantic parser within the domain of natural language queries to databases, explicitly devise quantifier scoping in the semantic model.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 64–72,
Sofia, Bulgaria, August 4-9 2013.

over the collection of entities introduced by the plural.[2] Scoping this implicit universal is just as important. While explicit universals may not occur very often in natural language, the usage of plurals is very common. Plurals form 18% of the NPs in our corpus and 20% of the nouns in Penn Treebank. Explicit universals, on the other hand, form less than 1% of the determiners in Penn Treebank. Quantifiers are also affected by negation. Previous work (e.g., Morante and Blanco, 2012) has investigated automatically detecting the scope and focus of negation. However, the scope of negation with respect to quantifiers is a different phenomenon. Consider the following sentence.

2. *The word does not start with a capital letter.*

Transforming this sentence into a meaning representation language, for almost any practical purposes, requires deciding whether the NP *a capital letter* lies in the scope of the negation or outside of it. The former describes the preferred reading where *The word* starts with a lowercase letter as in *apple, orange, banana*, but the latter gives the unlikely reading, according to which there exists a particular *capital letter*, say *A*, that *The word* starts with, as in *apple, Orange, Banana*. By not involving negation in quantifier scoping, a semantic parser may produce an unintended interpretation.

Previous work on statistical QSD has been quite restricted. Higgins and Sadock (2003), which we refer to as **HS03**, developed the first statistical QSD system for English. Their system disambiguates the scope of exactly two explicitly quantified NPs in a sentence, ignoring indefinite *a/an*, definites and bare NPs. Manshadi and Allen (2011a), hence **MA11**, go beyond those limitations and scope an arbitrary number of NPs in a sentence with no restriction on the type of quantification. However, although their corpus annotates the scope of negations and the implicit universal of plurals, their QSD system does not handle those.

As a step towards comprehensive automatic QSD, in this paper we present our work on automatic scoping of the implicit universal of plurals and negations. For data, we use a new revision of MA11's corpus, first introduced in Manshadi et al. (2011b). The new revision, called **QuanText**, carries a more detailed, fine-grained scope annotation (Manshadi et al., 2012). The performance of our model defines a baseline for future efforts on (comprehensive) QSD over QuanText. In addition to addressing plurality and negation, this work improves upon MA11's in two directions.

- We theoretically justify MA11's ternary-classification approach, formulating it as a general framework for learning to build partial orders. An $n \log n$ algorithm is then given to find a guaranteed approximation within a fixed ratio of the optimal solution from a set of pairwise preferences (Sect. 3.1).
- We replace MA11's hand-annotated features with a set of automatically generated linguistic features. Our rich set of features significantly improves the performance of the QSD model, even though we give up the gold-standard dependency features (Sect. 3.3).

2 Task definition

In QuanText, scope-bearing elements (or, as we call them, scopal terms) of each sentence have been identified using labeled **chunks**, as in (3).

3. *Replace **[1/** every line**]** in **[2/** the file**]** ending in **[3/** punctuation**]** with **[4/** a blank line**]** .*

NP chunks follow the definition of **baseNP** (Ramshaw and Marcus, 1995) and hence are flat. Outscoping relations are used to specify the relative scope of scopal terms. The relation $i > j$ means that chunk i **outscopes** (or has **wide scope** over) chunk j. Equivalently, chunk j is said to have **narrow scope** with respect to i. Each sentence is annotated with its most preferred scoping (according to the annotators' judgement), represented as a **partial order**:

4. $SI : (2 > 1 > 4;\ 1 > 3)$

If neither $i > j$ nor $j > i$ is entailed from the scoping, i and j are **incomparable**. This happens if both orders are equivalent (as in two existentials) or when the two chunks have no scope interaction.

Since a partial order can be represented by a **Directed Acyclic Graph** (DAG), we use DAGs to represent scopings. For example, G_1 in Figure 1 represents the scoping in (4).

2.1 Evaluation metrics

Given the gold standard DAG $G_g = (V, E_g)$ and the predicted DAG $G_p = (V, E_p)$, a similarity measure may be defined based on the ratio of the number of pairs (of nodes) labeled correctly to the

[2] Although plurals carry different types of quantification (Herbelot and Copestake, 2010), almost always there exists an implicit universal. The importance of scoping this universal, however, may vary based on the type of quantification.

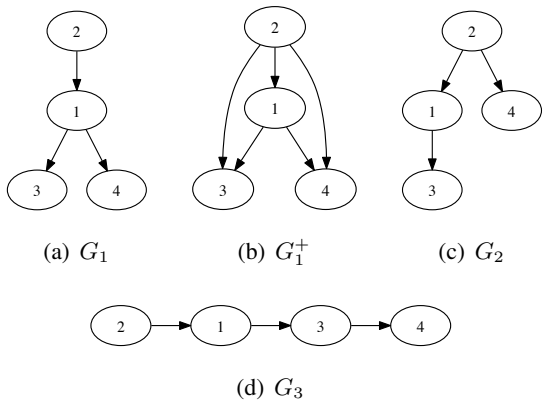

Figure 1: Scoping as DAG

total number of pairs. In order to take the transitivity of outscoping relations into account, we use the **transitive closure** (TC) of DAGs. Let $G^+ = (V, E^+)$ represent the TC of a DAG $G = (V, E)$.[3] G_1 and G_1^+ in Figure 1 illustrate this concept. We now define the similiarty metric S^+ as follows:

$$\sigma^+ = \frac{|E_p^+ \cap E_g^+| \cup |\bar{E_p^+} \cap \bar{E_g^+}|}{|V|(|V|-1)/2} \quad (1)$$

in which $\bar{G} = (V, \bar{E})$ is the complement of the underlying undirected version of G.

HS03 and others have used such a similarity measure for evaluation purposes. A disadvantage of this metric is that it gives the same weight to outscoping and incomparability relations. In practice, if two scopal terms with equivalent ordering (and hence, no outscoping relation) are incorrectly labeled with an outscoping, the logical form still remains valid. But if an outscoping relation is mislabeled, it will change the interpretation of the sentence. Therefore, in MA11, we suggest defining a precision/recall based on the number of outscoping relations recovered correctly: [4]

$$P^+ = \frac{|E_p^+ \cap E_g^+|}{|E_p^+|}\ ,\ R^+ = \frac{|E_p^+ \cap E_g^+|}{|E_g^+|} \quad (2)$$

[3] $(u,v) \in G^+ \iff ((u,v) \in G \vee \exists w_1 \ldots w_n \in V, (u,w_1) \ldots (w_n,v) \in E)$

[4] MA11 argues that TC-based metrics tend to produce higher numbers. For example if G_3 in Figure 1 is a gold-standard DAG and G_1 is a candidate DAG, TC-based metrics count $2 > 3$ as another match, even though it is entailed from $2 > 1$ and $1 > 3$. They give an alternative metric based on **transitive reduction** (TR), obtained by removing all the redundant edges of a DAG. TR-based metrics, however, have their own disadvantage. For example, if G_2 is another candidate for G_3, TR-based metrics produce the same numbers for both G_1 and G_2, even though G_1 is clearly closer to G_3 than G_2. Therefore, in this paper we stick to TC-based metrics.

3 Our framework

3.1 Learning to do QSD

Since we defined QSD as a partial ordering, automatic QSD would become the problem of learning to build partial orders. The machine learning community has studied the problem of learning **total orders (ranking)** in depth (Cohen et al., 1999; Furnkranz and Hullermeier, 2003; Hullermeier et al., 2008). Many ranking systems create partial orders as output when the confidence level for the relative order of two objects is below some threshold. However, the target being a partial order is a fundamentally different problem. While the lack of order between two elements is interpreted as the lack of confidence in the former, it should be interpreted as **incomparability** in the latter. Learning to build partial orders has not attracted much attention in the learning community, although as seen shortly, the techniques developed for ranking can be adopted for learning to build partial orders.

As mentioned before, a partial order P can be represented by a DAG G, with a preceding b in P if and only if a reaches b in G by a directed path. Although there could be many DAGs representing a partial order P, only one of those is a **transitive** DAG.[5] Therefore, in order to have a one-to-one relationship between QSDs and DAGs, we only consider the class of transitive DAGs, or **TDAG**. Every non-transitive DAG will be converted into its transitive counterpart by taking its transitive closure (as shown in Figure 1).

Consider V, a set of nodes and a TDAG $G = (V, E)$. It would help to think of disconnected nodes u, v of G, as connected with a null edge ϵ. We define the labeling function $\delta_G : V \times V \longrightarrow \{+, -, \epsilon\}$ assigning one of the three labels to each pair of nodes in G:

$$\delta_G(u,v) = \begin{cases} + & (u,v) \in G \\ - & (v,u) \in G \\ \epsilon & \text{otherwise} \end{cases} \quad (3)$$

Given the true TDAG $\hat{G} = (V, \hat{E})$, and a candidate TDAG G, we define the **Loss** function to be the total number of incorrect edges:

$$L(G, \hat{G}) = \sum_{u \prec v \in V} I(\delta_G(u,v) \neq \delta_{\hat{G}}(u,v)) \quad (4)$$

in which $\prec$ is an arbitrary total order over the nodes in V[6], and $I(\cdot)$ is the indicator function. We

[5] G is transitive iff $(u,v), (v,w) \in G \Longrightarrow (u,w) \in G$.

[6] E.g., the left-to-right order of the corresponding chunks in the sentence.

adopt a **minimum Bayes risk** (MBR) approach, with the goal of finding the graph with the lowest expected loss against the (unknown) target graph:

$$G^* = \operatorname*{argmin}_{G \in TDAG} E_{\hat{G}}\left[L(G, \hat{G})\right] \quad (5)$$

Substituting in the definition of the loss function and exchanging the order of the expectation and summation, we get:

$$\begin{aligned} G^* &= \operatorname*{argmin}_{G \in TDAG} \sum_{u \prec v \in V} E_{\hat{G}}\left[I(\delta_G(u,v) \neq \delta_{\hat{G}}(u,v)\right] \\ &= \operatorname*{argmin}_{G \in TDAG} \sum_{u \prec v \in V} P(\delta_G(u,v) \neq \delta_{\hat{G}}(u,v)) \quad (6) \end{aligned}$$

This means that in order to solve Eq. (5), we need only the probabilities of each of the three labels for each of the $C(n,2) = n(n-1)/2$ pairs of nodes[7] in the graph, rather than a probability for each of the superexponentially many possible graphs. We train a classifier to estimate these probabilities directly for a given pair. Therefore, we have reduced the problem of predicting a partial order to **pairwise comparison**, analogous to ranking by pairwise comparison or **RPC** (Hullermeier et al., 2008; Furnkranz and Hullermeier, 2003), a popular technique in learning total orders. The difference though is that in RPC, the comparison is a (soft) binary classification, while for partial orders we have the case of incomparability (the label ϵ), hence a (soft) ternary classification.

A soft ternary classifier generates three probabilities, $p_{u,v}(+)$, $p_{u,v}(-)$, and $p_{u,v}(\epsilon)$ for each pair (u,v),[8] corresponding to the three labels. Hence, equation Eq. (6) can be rearranged as follows:

$$G^* = \operatorname*{argmax}_{G \in TDAG} \sum_{u \prec v \in V} p_{u,v}(\delta_G(u,v)) \quad (7)$$

Let Γ_p be a graph like the one in Figure 2, containing exactly three edges between every two nodes, weighted by the probabilities from the $n(n-1)/2$ classifiers. We call Γ_p the **preference graph**. Intuitively speaking, the solution to Eq. (7) is the transitive directed acyclic subgraph of Γ_p that has the maximum sum of weights. Unfortunately finding this subgraph is an NP-hard problem.[9]

[7] Throughout this subsection, unless otherwise specified, by a pair of nodes we mean a pair (u,v) with $u \prec v$.

[8] $p_{v,u}$ for $u \prec v$ is defined in the obvious way: $p_{v,u}(+) = p_{u,v}(-)$, $p_{v,u}(-) = p_{u,v}(+)$, and $p_{v,u}(\epsilon) = p_{u,v}(\epsilon)$.

[9] The proof is beyond the scope of this paper, but the idea is similar to that of Cohen et al. (1999), on finding total orders. Although they don't use an RPC technique, Cohen et al. (1999) encounter a similar optimization problem. They propose an approximation algorithm which finds the solution (a total order) in a greedy way. Here we use the same greedy technique to find a total order, but take it only as the topological order of the solution (Figure 3).

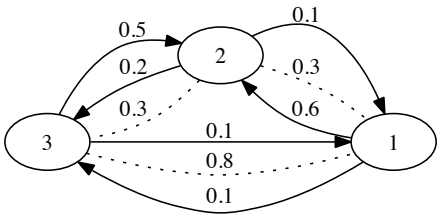

Figure 2: A preference graph over three nodes.

1. Let Γ_p be the preference graph and set G to $\emptyset$.
2. $\forall u \in V$, let $\pi(u) = \sum_v p_{u,v}(+) - \sum_v p_{u,v}(-)$.
3. Let $u^* = \operatorname{argmax}_u \pi(u)$, $S^- = \sum_{v \in G} p_{v,u^*}(-)$ & $S^\epsilon = \sum_{v \in G} p_{v,u^*}(\epsilon)$.
4. Remove u^* and all its incident edges from Γ_p.
5. Add u^* to G; also if $S^- > S^\epsilon$, for every $v \in G - u^*$, add (v, u^*) to G.
6. If Γ_p is empty, output G, otherwise repeat steps 2-5.

Figure 3: An approximation algorithm for Eq. (7)

Since it is very unlikely to find an efficient algorithm to solve Eq. (7), instead, we propose the algorithm in Figure 3 which finds an approximate solution. The idea of the algorithm is simple. By finding u^* with the highest $\pi(u)$ in step 3, we form a **topological order** for the nodes in G in a greedy way (see Footnote 9). We then add u^* to G. A directed edge is added either from every node in $G - u^*$ to u^* or from no node, depending on which case makes the sum of the weights in G higher.

Theorem 1 *The algorithm in Figure 3 is a 1/3-OPT approximation algorithm for Eq. (7).*

Proof idea. First of all, note that G is a TDAG, because edges are only added to the most recently created node in step 5. Let OPT be the optimum value of the right hand side of Eq. (7). The sum of all the weights in Γ_p is an upper bound for OPT:

$$\sum_{u \prec v \in V} \sum_{\lambda \in \{+,-,\epsilon\}} p_{u,v}(\lambda) \geq OPT$$

Step 5 of the algorithm guarantees that the labels $\delta_G(u,v)$ satisfy:

$$\sum_{u \prec v \in V} p_{u,v}(\delta_G(u,v)) \geq \sum_{u \prec v \in V} p_{u,v}(\lambda) \quad (8)$$

for any $\lambda \in \{+, -, \epsilon\}$. Hence:

$$\sum_{u \prec v \in V} p_{u,v}(\delta_G(u,v)) = \frac{1}{3}\left(3 \sum_{u \prec v \in V} p_{u,v}(\delta_G(u,v))\right)$$
$$\geq \frac{1}{3} \sum_{u \prec v \in V} \sum_{\lambda \in \{+,-,\epsilon\}} p_{u,v}(\lambda)$$
$$\geq \frac{1}{3} OPT$$

In practice, we improve the algorithm in Figure 3, while maintaining the approximation guarantee, as follows. When adding a node u^* to graph G, we do not make a binary decision as to whether connect every node in G to u^* or none, but we use some heuristics to choose a subset of nodes (possibly empty) in G that if connected to u^* results in a TDAG whose sum of weights is at least as big as the binary none-vs-all case. As described in Sec. 4, the algorithm works very well in our QSD system, finding the optimum solution in virtually all cases we examined.

3.2 Dealing with plurality and negation

Consider the following sentence with the plural NP chunk *the lines*.

5. *Merge **[1p/** the lines], ending in **[2/** a punctuation], with **[3/** the next non-blank line].*
6. $SI : (1c > 1d > 2;\ 1d > 3)$ [10]

In QuanText, plural chunks are indexed with a number followed by the lowercase letter "p". As seen in (6), the scoping looks different from before in that the terms $1d$ and $1c$ are not the label of any chunk. These two terms refer to the two quantified terms introduced by the plural chunk $1p$: $1c$ (for **collection**) represents the set (or in better words collection) of entities, defined by the plural, and $1d$ (for **distribution**) refers to the implicit universal, introduced by the plural. In other words, for a plural chunk ip, id represents the universally quantified entity over the collection ic. The outscoping relation $1d > 2$ in (6) states that every *line* in the collection, denoted by $1c$, starts with its own *punctuation* character. Similarly, $1d > 3$ indicates that *every line* has its own *next non-blank line*. Figure 4(a) shows a DAG for the scoping in (6).

In (7) we have a sentence containing a negation. In QuanText, negation chunks are labeled with an uppercase "N" followed by a number.

[10] This scoping corresponds to the logical formula:
$\mathcal{D}x_{1c}, Collection(x_{1c}) \wedge \forall x_{1d}, In(x_{1d}, x_{1c}) \Rightarrow (Line(x_{1d}) \wedge (\exists x_2, Punctuation(x_2) \wedge EndIn(x_{1d}, x_2)) \wedge (\mathcal{D}x_3, \neg blank(x_3) \wedge next(x_{1d}, x_3) \wedge merge(x_{1d}, x_3)))$
It is straightforward to write a formula for, say, $1c > 2 > 1d$.

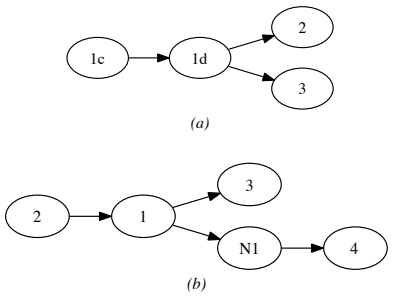

Figure 4: DAGs for scopings in (6) and (8)

7. *Extract **[1/** every word] in **[2/** file "1.txt"], which starts with **[3/** a capital letter], but does **[N1/** not] end with **[4/** a capital letter].*
8. $SI : (2 > 1 > 3;\ 1 > N1 > 4)$

As seen here, a negation simply introduces a chunk, which participates in outscoping relations like an NP chunk. Figure 4(b) represents the scoping in (8) as a DAG.

From these examples, as long as we create two nodes in the DAG corresponding to each plural chunk, and one node corresponding to each negation, there is no need to modify the underlying model (defined in the previous section). However, when u (or v) is a negation (Ni) or an implicit universal (id) node, the probabilities $p^{\lambda}_{u,v}$ ($\lambda \in \{+, -, \epsilon\}$) may come from a different source, e.g. a different classification model or the same model with a different set of features, as described in the following section.

3.3 Feature selection

Previous work has shown that the lexical item of quantifiers and syntactic clues (often extracted from phrase structure trees) are good at predicting quantifier scoping. Srinivasan and Yates (2009) use the semantics of the head noun in a quantified NP to predict the scoping. MA11 also find the lexical item of the head noun to be a good predictor. In this paper, we introduce a new set of syntactic features which we found very informative: the "type" dependency features of de Marneffe et al. (2006). Adopting this new set of features, we outperform MA11's system by a large margin. Another point to mention here is that the features that are predictive of the relative scope of quantifiers are not necessarily as helpful when determining the scope of negation and vice versa. Therefore we do not use exactly the same set of features when

one of the scopal terms in the pair[11] is a negation, although most of the features are quite similar.

3.3.1 NP chunks

We first describe the set of features we have adopted when both scopal terms in a pair are NP-chunks. We have organized the features into different categories listed below.

Individual NP-chunk features

Following features are extracted for both NP chunks in a pair.

- The part-of-speech (POS) tag of the head of chunk
- The lexical item of the head noun
- The lexical item of the determiner/quantifier
- The lexical item of the pre-determiner
- Does the chunk contain a constant (e.g. "do", 'x')?
- Is the NP-chunk a plural?

Implicit universal of a plural

Remember that every plural chunk i introduces two nodes in the DAG, ic and id. Both nodes are introduced by the same chunk i, therefore they use the same set of features. The only exception is a single additional binary feature for plural NP chunks, which determines whether the given node refers to the implicit universal of the plural (i.e. id) or to the collection itself (i.e. ic).

- Does this node refer to an implicit universal?

Syntactic features – phrase structure tree

As mentioned above, we have used two sets of syntactic features. The first is motivated by HS03's work and is based on the constituency (i.e. phrase structure) tree T of the sentence. Since our model is based on pairwise comparison, the following features are defined for each pair of chunks. In the following, by chunk we mean the deepest phrase-level node in T dominating all the words in the chunk. If the constituency tree is correct, this node is usually an NP node. Also, P refers to the undirected path in T connecting the two chunks.

- Syntactic category of the deepest common ancestor
- Does 1st/2nd chunk *C-command* 2nd/1st one?
- Length of the path P
- Syntactic categories of nodes on P
- Is there a conjoined node on P?
- List of punctuation marks dominated by nodes on P

Syntactic features – dependency tree

Although regular "untyped" dependency relations do not seem to help our QSD system in the presence of phrase-structure trees, we found the *collapsed typed dependencies* (de Marneffe and Manning, 2008) very helpful, even when used on top of the phrase-structure features. Below is the list of features we extract from the collapsed typed dependency tree T_d of each sentence. In the following, by noun we mean the node in T_d which corresponds to the head of the chunk. The choice of the word *noun*, however, may be sloppy, as the head of an NP chunk may not be a noun.

- Does 1st/2nd noun dominate 2nd/1st noun?
- Does 1st/2nd noun immediately dominate 2nd/1st?
- Type of incoming dependency relation of each noun
- Syntactic category of the deepest common ancestor
- Lexical item of the deepest common ancestor
- Length of the undirected path between the two

3.3.2 Negations

There are no sentences in our corpus with more than one negation. Therefore, for every pair of nodes with one negation, the other node must refer to an NP chunk. We use the following word-level, phrase-structure, and dependency features for these pairs.

- Lexical item of the determiner for the NP chunk
- Does the NP chunk contain a constant?
- Is the NP chunk a plural?
- If so, does this node refer to its implicit universal?
- Does the negation C-command the NP chunk in T?
- Does the NP chunk C-command the negation in T?
- What is the POS of the parent p of negation in T_d?
- Does p dominate the noun in T_d?
- Does the noun dominate p in T_d?
- Does p immediately dominate the noun in T_d?
- If so, what is the type of the dependency?
- Does the noun immediately dominate p in T_d?
- If so, what is the type of the dependency?
- Length of the undirected path between the two in T_d

4 Experiments

QuanText contains 500 sentences with a total of 1750 chunks, that is 3.5 chunks/sentence on average. Of those, 1700 chunks are NP chunks. The rest are scopal operators, mainly negation. Of all the NP chunks, 320 (more than 18%) are plural, each introducing an implicit universal, that is, an additional node in the DAG. Since we feed each pair of elements to the classifiers independently, each (unordered) pair introduces one sample. Therefore, a sentence with n scopal elements creates $C(n, 2) = n(n-1)/2$ samples for classification. When all the elements are taken into account,[12] the total number of samples in the corpus will be:

[11] Since our model is based on pairwise comparison, every sample is in fact a pair of nodes (u, v) of the DAG.

[12] Here by all elements we mean explicit chunks and the implicit universals. QuanText labels some other (implicit) elements, which we have not been handled in this work. In particular, some nouns introduce two entities: a **type** and a

$$\sum_i C(n_i, 2) \approx 4500 \quad (9)$$

Where n_i is the number of scopal terms introduced by sentence i. Out of the 4500 samples, around 1800 involve at least one implicit universal (i.e., id), but only 120 samples contain a negation. We evaluate the performance of the system for implicit universals and negation both separately and in the context of full scope disambiguation. We split the corpus at random into three sets of 50, 100, and 350 sentences, as development, test, and train sets respectively.[13]

To extract part-of-speech tags, phrase structure trees, and typed dependencies, we use the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006) on both train and test sets. Since we are using SVM, we have passed the confidence levels through a softmax function to convert them into probabilities $P^{\lambda}_{u,v}$ before applying the algorithm of Section 3. We take MA11's system as the baseline. However, in order to have a fair comparison, we have used the output of the Stanford parser to automatically generate the same features that MA11 have hand-annotated.[14] In order to run the baseline system on implicit universals, we take the feature vector of a plural NP and add a feature to indicate that this feature vector represents the implicit universal of the corresponding chunk. Similarly, for negation we add a feature to show that the chunk represents a negation. As shown in Section 3.3.2, we have used a more compact set of features for negations. Once again, in order to have a fair comparison, we apply a similar modification to the baseline system. We also use the exact same classifier as used in MA11.[15] Figure 5(a) compares the performance of our model, which we refer to as RPC-SVM-13, with the baseline system, but only on explicit NP chunks.[16] The goal for running this experiment has been to compare the performance of our model to the baseline system on the task that it was actually defined to perform (that is scoping only explicit NP chunks).

NP-Chunks only (no *id* or negation)	σ^+	P^+	R^+	F^+	AR	A
Baseline (MA11)	0.762	0.638	0.484	0.550	0.59	0.47
Our model (RPC-SVM-13)	0.827	0.743	0.677	0.709	0.68	0.55

(a) Scoping explicit NP chunks

Overall system (including negation and implicit universals)	σ^+	P^+	R^+	F^+	AR	A
Baseline (MA11)	0.787	0.688	0.469	0.557	0.59	0.47
Our model (RPC-SVM-13)	0.863	0.784	0.720	0.751	0.69	0.55

(b) Scoping all elements (including id and Ni)

Figure 5: Performance on QuanText data

As seen in this table, by incorporating a richer set of features and a better learning algorithm, our model outperforms the baseline by almost 15%. The measure A in these figures shows sentence-based accuracy. A sentence counts as correct iff every pair of scopal elements has been labeled correctly. Therefore A is a tough measure. Furthermore, it is sensitive to the length of the sentence. Following MA11, we have computed another sentence-based accuracy measure, AR. In computing AR, a sentence counts as correct iff all the outscoping relations have been recovered correctly – in other words, iff $R = 100\%$, regardless of the value of P. AR may be more practically meaningful, because if in the correct scoping of the sentence there is no outscoping between two elements, inserting one does not affect the interpretation of the sentence. In other words, precision is less important for QSD in practice.

Figure 5(b) gives the performance of the overall model when all the elements including the implicit universals and the negations are taken into account. That the F-score of our model for the second experiment is 0.042 higher than F-score for the first indicates that scoping implicit universals and/or negations must be easier than scoping explicit NP chunks. In order to find how much one or both of the two elements contribute to this gain, we have run two more experiments, scoping only the pairs with at least one implicit universal and pairs with one negation, respectively. Figure 6 reports the results. As seen, the contribution in boosting the overall performance comes from the implicit universals while negations, in fact, lower the performance. The performance for pairs with implicit universal is higher because universals, in general,

token, as described by Manshadi et al. (2012). In this work, we have only considered the token entity introduced by those nouns and have ignored the type entity.

[13] Since the percentage of sentences with negation is small, we made sure that those sentences are distributed uniformly between three sets.

[14] MA11's features are similar to part-of-speech tags and untyped dependency relations.

[15] $SVM^{Multiclass}$ from *SVM-light* (Joachims, 1999).

[16] In all experiments, we ignore NP conjunctions. Previous work treats a conjunction of NPs as separate NPs. However, similar to plurals, NP conjunctions (disjunctions) introduce an extra scopal element: a universal (existential). We are working on an annotation scheme for NP conjunctions, so we have left this for after the annotations become available.

Implicit universals only (pairs with at least one *id)*	P^+	R^+	F^+
Baseline (MA11)	0.776	0.458	0.576
Our model (RPC-SVM-13)	0.836	0.734	0.782

(a) Pairs with at least one implicit universal

Negation only (pairs with one negation)	P^+	R^+	F^+
Baseline (MA11)	0.502	0.571	0.534
Our model (RPC-SVM-13)	0.733	0.55	0.629

(b) Pairs with at least one negation

Figure 6: Implicit universals and negations

are easier to scope, even for the human annotators.[17] There are several reasons for poor performance with negations as well. First, the number of negations in the corpus is small, therefore the data is very sparse. Second, the RPC model does not work well for negations. Scoping a negation relative to an NP chunk, with which it has a long distance dependency, often depends on the scope of the elements in between. Third, scoping negation usually requires a deep semantic analysis.

In order to see how well our approximation algorithm is working, similar to the approach of Chambers and Jurafsky (2008), we tried an ILP solver[18] for DAGs with at most 8 nodes to find the optimum solution, but we found the difference insignificant. In fact, the approximation algorithm finds the optimum solution in all but one case.[19]

5 Related work

Since automatic QSD is in general challenging, traditionally quantifier scoping is left underspecified in deep linguistic processing systems (Alshawi and Crouch, 1992; Bos, 1996; Copestake et al., 2001). Some efforts have been made to move underspecification frameworks towards weighted constraint-based graphs in order to produce the most preferred reading (Koller et al., 2008), but the source of these types of constraint are often discourse, pragmatics, world knowledge, etc., and hence, they are hard to obtain automatically. In order to evade scope disambiguation, yet be able to perform entailment, Koller and Thater (2010) propose an algorithm to calculate the weakest readings[20] from a scope-underspecified representation.

Early efforts on automatic QSD (Moran, 1988; Hurum, 1988) were based on heuristics, manually formed into rules with manually assigned weights for resolving conflicts. To the best of our knowledge, there have been four major efforts on statistical QSD for English: Higgins and Sadock (2003), Galen and MacCartney (2004), Srinivasan and Yates (2009), and Manshadi and Allen (2011a). The first three only scope two scopal terms in a sentence, where the scopal term is an NP with an explicit quantification. MA11 is the first to scope any number of NPs in a sentence with no restriction on the type of quantification. Besides ignoring negation and implicit universals, their system has some other limitations too. First, the learning model is not theoretically justified. Second, hand-annotated features (e.g. dependency relations) are used on both the train and the test data.

6 Summary and future work

We develop the first statistical QSD model addressing the interaction of quantifiers with negation and the implicit universal of plurals, defining a baseline for this task on QuanText data (Manshadi et al., 2012). In addition, our work improves upon Manshadi and Allen (2011a)'s work by (approximately) optimizing a well justified criterion, by using automatically generated features instead of hand-annotated dependencies, and by boosting the performance by a large margin with the help of a rich feature vector.

This work can be improved in many directions, among which are scoping more elements such as other scopal operators and implicit entities, deploying more complex learning models, and developing models which require less supervision.

Acknowledgement

We need to thank William de Beaumont and Jonathan Gordon for their comments on the paper and Omid Bakhshandeh for his assistance. This work was supported in part by NSF grant 1012205, and ONR grant N000141110417.

[17] Trivially, we have taken the relation outscoping $ic > id$ for granted and not counted it towards higher performance.

[18] lpsolve: http://sourceforge.net/projects/lpsolve

[19] To find the gain that can be obtained with gold-standard parses, we used MA11's system with their hand-annotated and the equivalent automatically generated features. The former boost the performance by 0.04. Incidentally, HS03 lose almost 0.04 when switching to automatically generated parses.

[20] Those which can be entailed from other readings but do not entail any other reading

References

Hiyan Alshawi and Richard Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of Association for Computational Linguistics*, pages 32–39.

Johan Bos. 1996. Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–143.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 698–706, Stroudsburg, PA.

William W. Cohen, Robert E. Schapire, and Yoram Singer. 1999. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of Association for Computational Linguistics '01*, pages 140–147.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure trees. In *Proceedings of International Conference on Language Resources and Evaluation '12*.

Johannes Furnkranz and Eyke Hullermeier. 2003. Pairwise preference learning and ranking. In *Proceedings of the 14th European Conference on Machine Learning*, volume 2837, pages 145–156.

Andrew Galen and Bill MacCartney. 2004. Statistical resolution of scope ambiguity in natural language. http://nlp.stanford.edu/nlkr/scoper.pdf.

Fritz Hamm and Edward W. Hinrichs. 2010. *Plurality and Quantification*. Studies in Linguistics and Philosophy. Springer.

Aurelie Herbelot and Ann Copestake. 2010. Annotating underquantification. In *Proceedings of the Fourth Linguistic Annotation Workshop*, LAW IV '10, pages 73–81.

Derrick Higgins and Jerrold M. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1):73–96.

Eyke Hullermeier, Johannes Furnkranz, Weiwei Cheng, and Klaus Brinker. 2008. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(1617):1897 – 1916.

Sven Hurum. 1988. Handling scope ambiguities in English. In *Proceedings of the second conference on Applied natural language processing*, ANLC '88, pages 58–65.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in kernel methods*, pages 169–184. MIT Press, Cambridge, MA, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430.

Alexander Koller and Stefan Thater. 2010. Computing weakest readings. In *Proceedings of the 48th Annual Meeting on Association for Computational Linguistics*, Uppsala, Sweden.

Alexander Koller, Michaela Regneri, and Stefan Thater. 2008. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of Annual Meeting on Association for Computational Linguistics and Human Language Technologies '08*.

Fred Landmann. 2000. *Events and plurality*. Kluwer Academic Publishers, Dordrecht.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of Association for Computational Linguistics (ACL)*.

Mehdi Manshadi and James Allen. 2011a. Unrestricted quantifier scope disambiguation. In *Proceedings of Association for Computational Linguistics '11, Workshop on Graph-based Methods for NLP (TextGraph-6)*.

Mehdi Manshadi, James Allen, and Mary Swift. 2011b. A corpus of scope-disambiguated English text. In *Proceedings of Association for Computational Linguistics and Human Language Technologies '11: short papers*, pages 141–146.

Mehdi Manshadi, James Allen, and Mary Swift. 2012. An annotation scheme for quantifier scope disambiguation. In *Proceedings of International Conference on Language Resources and Evaluation '12*.

Douglas Moran. 1988. Quantifier scoping in the SRI core language engine. In *Proceedings of the 26th Annual Meeting on Association for Computational Linguistics*.

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey.

Prakash Srinivasan and Alexander Yates. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: preliminary results. In *Proceedings of EMNLP '09*.