An Annotation Scheme for Quantifier Scope Disambiguation

Mehdi Manshadi, James Allen, Mary Swift

Department of Computer Science, University of Rochester
Rochester, NY 14627
{mehdih,james,swift}@cs.rochester.edu

Abstract

Annotating natural language sentences with quantifier scoping has proved to be very hard. In order to overcome the challenge, previous work on building scope-annotated corpora has focused on sentences with two explicitly quantified noun phrases (NPs). Furthermore, it does not address the annotation of scopal operators or complex NPs such as plurals and definites. We present the first annotation scheme for quantifier scope disambiguation where there is no restriction on the type or the number of scope-bearing elements in the sentence. We discuss some of the most prominent complex scope phenomena encountered in annotating the corpus, such as plurality and type-token distinction, and present mechanisms to handle those phenomena.

Keywords: Quantifier Scoping, Annotation Scheme, Scope Disambiguation

1. Introduction

There are two main reasons for the lack of work on automatic quantifier scope disambiguation. First, it is generally believed that instances of scope ambiguities are rare. The rich literature on quantifier scope ambiguity in theoretical semantics and philosophy has probably contributed to this misunderstanding. Researchers in those areas are mainly interested in situations where there are in fact multiple plausible readings, as in "Three men carried four tables". Those situations do not happen very often in the daily usage of natural language, and when they do, we may not care to get the exact scoping of the sentence, and even if we do, it is fine in those rare cases to stop the speaker and ask a follow-up question in order to resolve the ambiguity! In computational semantics, however, every sentence with more than one scope-bearing element carries scope ambiguity.

Recent work on semantic parsing (Liang et al., 2011) shows that even in a very restricted domain such as answering questions about the geography of the US (where most NPs are expected to be proper nouns and hence carry no scope ambiguity), addressing quantifier scoping will improve the performance. We believe that as NLP systems start to go deeper in language understanding, quantifier scope ambiguity will become a bigger issue.

Second, no corpus has been provided with full scope disambiguation. With no corpus at hand, even developing an unsupervised model seems challenging, as there are no available test sets to evaluate the performance of the model (Srinivasan and Yates, 2009). The lack of labeled corpora is because the hand annotation of quantifier scoping is surprisingly hard. As a result, previous work (Higgins and Sadock, 2003; Galen and MacCartney, 2004) has focused on annotating sentences with two explicitly quantified NPs as the explicit quantification makes the scoping more intuitive.[1] The 52% inter-annotator agreement (IAA) achieved by Higgins and Sadock (2003) on a simpler task is evidence that this kind of annotation is difficult.

As mentioned in Manshadi et al. (2011), in order to overcome these challenges, we have chosen a domain in which a conscious knowledge of scoping is often required in order to fully understand a sentence and be able to accomplish the requested task. Within this domain, however, we do not put any restriction on the number of scopal elements in the sentence. We examine every two NPs (including definites, indefinites, bare nouns, etc.) for possible scope interactions. Furthermore we scope almost all *scopal operators* (a.k.a. *fixed-scopals*; c.f. Copestake et al. (2005)).

We have had two goals in mind for developing the annotation scheme. First, we have worked toward addressing complex scope phenomena (e.g. plurality) by devising the necessary machinery to represent the full scoping. Second we have targeted achieving a high quality annotation with high IAA by isolating the scope disambiguation task from other semantic phenomena, which are sometimes mixed up with quantifier scope ambiguity (the latter was proved to be a major source of disagreement in our early experiments). Manshadi et al. (2011) describes the general properties of our corpus. In this paper, we describe the basics of our annotation scheme and the notation we have invented to represent scope disambiguations. There are some technical and theoretical details in our scheme that are beyond the scope of this paper. Those are left for a longer version.

The structure of the paper is as follows. An overview of our domain is given in Section 2. We chunk the sentences in this domain into scope-bearing elements. The process is discussed in Section 3. The core of the annotation scheme and the notation is presented in Section 4. In Section 5 and 6 we discuss the two most challenging phenomena encountered in the process of scope annotation and extend our notation to address them. Section7 converts scope annotations into a graph representation for evaluation purposes.

2. Data

Our domain of choice is natural language instructions on editing plain text files. We have picked this domain for the

[1] The usage of the word *explicit* here is sloppy and is meant to rule out definites, indefinites, and bare nouns. Throughout this paper, unless otherwise specified, by "quantifier" we mean "generalized quantifier" including definite, indefinite and bare NPs.

following reasons:

- The sentences in this domain contain a fair amount of explicit quantifiers creating scope interactions.
- Quantifier scoping is critical for such tasks. In order to be able to accomplish a task, the (human) agent must have a *conscious* knowledge of quantifier scoping, as the task requires (subconsciously) converting the natural language instructions into formal descriptions. This means that the scope annotation is fairly intuitive for human annotators.
- For the same reason, it is essential for a natural language understanding (NLU) system to figure out the scope disambiguation in order to accomplish the task. Therefore, while in some other domains an NLU system might work fine by leaving quantifier scoping underspecified (Allen et al., 2007; Bos, 2008), it would be hard to do well in this domain without having the scope ambiguity resolved in one way or another.

Our corpus consists of 500 sentences each describing a standalone task. For more information on the resources used to provide this corpus, and some statistics on the distribution of scopal elements in the corpus, see Manshadi et al. (2011).

3. Chunking

A natural way to build a scope-disambiguated corpus is to add a scope-disambiguation layer on top of an existing corpus labeled with scope-underspecified semantic representation, such as the Redwoods corpus (Oepen et al., 2002). As mentioned before scope annotation on such a broad-domain corpus is very difficult, hence we provided a corpus in a restricted domain (Section 2). Since we do not have the full (scope-underspecified) semantic representation of our corpus, we take a shortcut. We chunk every scope-bearing element in the sentence, and assign a unique ID to each chunk. We then use those IDs to represent the scopings as explained in the next sections. The chunks have been distinguished by being wrapped within square brackets. The chunk ID is placed right after the left square bracket and is immediately followed by a slash and a space, as shown in the following example.

1. *Cut [1/ Every line] in [2/ the file] ending in [3/ a comma] .*

There are three NPs in 1 labeled 1, 2, and 3. There are two major types of chunk, NP chunks and (scopal) operators.

3.1. NP chunks

NP chunks consist of all the shallow NP chunks. Our definition of shallow NP is the standard definition used in NP chunking systems, first introduced as *baseNP* by Ramshaw and Marcus (1995): "the initial portions of non-recursive noun phrases up to the head including determiners but not including postmodifying prepositional phrases or clauses". We chunk all the NPs including bare NPs and pronouns. This includes those NPs, which (sometimes controversially) act as an adverbial and introduce no first order entity in the domain of discourse. This is because we do not want to impose any presumption about the scoping of a sentence but to leave those decisions to the annotators. For example consider the sentence

2. *Print [1/ all the words] in [2/ the file] in [3/ order] .*

It may be argued that "order" creates an entity, the "order" of "the words" to be printed, which is unique with respect to all the words, hence has a wide scope. In contrast, one could argue that the preposition phrase (PP) "in order" behaves as an adverbial modifying the action of "printing", hence "order" does not participate in scoping. By chunking all the NPs while providing a mechanism for the annotators to pick either of the two alternatives, we do not force one of the above interpretations. Chunking is first done automatically and then revised manually, so it is gold standard. For details refer to Manshadi et al. (2011).

3.1.1. Pronouns and possessive determiners

Pronouns are chunked like any other NP, as are possessive determiners. However, since possessive determiners are by definition part of the shallow NPs, they create a nested chunk, as shown below:

3. *For [1/ each numerical field], delete [2/ [3/ its] decimal point]] followed by [4/ the subsequent digits] and [5/ the space] after [6/ it]] .*

WH-determiners and WH-pronouns, on the other hand, have not been chunked. The decision was made for two reasons. First, those words occur very rarely in our corpus. Second, even when they do , their scoping does not give any extra information about the scoping of the overall sentence. This is because unlike ordinary pronouns, WH-pronouns occur in limited syntactic structures. This strategy may be revised in future versions of the annotation scheme, if compelling arguments are found in favor of the other alternative.

3.1.2. NP Conjunctions

NP Conjunctions could simply be treated as distinct NPs and have their own chunks. However, almost always, the sub-NPs in an NP conjunction have the same scope level, and it is a painful job to repeat the scope constraints for each sub-NP. An alternative is to chunk them all as one big chunk, but this imposes the presumption that all sub-NPs always have the same level of scoping. Since we do not want to make any speculation, we choose a third option.

4. *Delete [1/ all the lines] ending in [2/ [2.1/ a comma], [2.2/ a semicolon], [2.3/ a question mark], [2.4/ an exclamation mark] or [2.5/ a period]] .*

We let every sub-NP have its own chunk indexed as 2.1, 2.2, etc., but at the same we allow annotators to refer to the whole conjunction by a single ID (2 in the above example). As a result, while providing the convenience of avoiding repetition of the same scope constraints for each individual sub-NP, we allow annotators to provide different scope constraints in exceptional cases. Note that we are not committed to providing such a convenient chunking for every NP conjunction. For example, for complex NP conjunctions, where the sub-NPs have post-modifiers, such chunking will be confusing, so it will be avoided.

5. *Delete [1/ all lines] ending in [2/ a word] with [3/ no uppercase letter] or [4/ a number] with [5/ no decimal points].*

3.1.3. Partitives

Consider the following sentence:

6. *Delete [1/ every line] in which [2/ some] of [3/ the fields] are numerical .*

7. *Print [1/ the first half] of [2/ the fields] in [3/ each row] .*

Some believe that "some of the fields" in 6 is a noun phrase with "some of the" being the determiner, and some believe that it is composed of two NPs. As shown in 6 we go with the second view. This is because we believe this phrase introduces two entities, a definite set "the fields" in "every line", and a subset of this set. Adopting this strategy may be better justified by looking at example 7. This example is less controversial, as it clearly introduces three entities: "each row", "the fields in each row", and "the first half" of those "fields". The two phrases have almost the same structure and we have not found any compelling argument why they should be treated in different ways.

3.2. Operator chunks

Operataor chunks include modal operators (e.g. "possibly", "probably", etc.), frequency adverbials (e.g. "twice"), sentential adverbials (e.g. conditionals, "while", "whenever", etc.), and negation. While the ID of NP chunks starts with numbers, the ID of scopal operators starts with a letter followed by a number. Negation starts with "N". Sentential operators start with "S" and the rest of the operators start with "O" as illustrated in the following examples.

8. *Cut [1/ every word] that occurs [O1/ twice] in [2/ a line] .*

9. *Erase [1/ every line] that does [N1/ not] contain [2/ a digit].*

10. *[S1/ If] there is [1/ a line] ending in [2/ a digit] , delete [3/ the last field] of [4/ it].*

4. Scope annotation: the basics

The chunked corpus is given to annotators to be labeled with scoping. The scope relation between every two chunks is either explicitly mentioned in the scoping or entailed from the rest of the relations as detailed later. If i, j are two arbitrary chunks in a sentence, the following four relations can be recognized between the two: *outscoping, scope-equvalence/no-interaction, coreference relation, bridging anaphora relation*. Trivially, coreference and bridging anaphora relations are only meaningful when both chunks are NP chunks. The scope annotation is a list of semicolon-delimited chains of relations wrapped within parentheses following the keyword "SI:" as shown below. Annotators may leave comments between "/*" and "*/". Therefore everything between these two tokens is ignored (i.e. is not interpreted as part of the scope representation).

11. $SI : (i_1\, \mathscr{R}_1\, i_2\, \mathscr{R}_2\, \ldots\ ;\ j_1\, \mathscr{P}_1\, j_2\, \mathscr{P}_2\, \ldots\ ;\ \cdots)$ */*The general structure of a scope representation*/*

Where $i_1, j_2, \ldots$ are chunk IDs and $\mathscr{R}_1, \mathscr{P}_2, ...$ are relations. In 11, in order to obtain a more concise (and often more readable) scoping, we have allowed relations to be cascaded. The following statement holds in general.

Statement 1 *Given a sentence with chunks i, j and k, $(i_1\, \mathscr{R}_1\, i_2\, \mathscr{R}_2\, i_3)$ is equivalent to $(i_1\, \mathscr{R}_1\, i_2\ ;\ i_2\, \mathscr{R}_2\, i_3)$.*

Annotators are allowed to provide more than one scoping for a sentence, with their most/least preferred reading being the first/last. In practice, an annotator very rarely labels a sentence with more than one scoping. The rest of this section describes all the relations in detail.

4.1. Outscoping constraints

An outscoping constraint is represented as $(i > j)$ denoting that chunk i has *wide scope over* or *outscopes* chunk j. For example, in sentence 1 (repeated in 12), we have $(2 > 1)$. This is because the "file" is fixed with respect to "every line". In other words, it is not the case that there is a distinct file corresponding to "every line". In contrast, there is a distinct "comma" that ends "every line". Therefore, "every line" has wide scope over "a comma", hence $(1 > 3)$.[2]

12. *Delete [1/ Every line] in [2/ the file] which ends with [3/ a comma] .*
$SI : (2 > 1\ ; 1 > 3)$

Trivially outscoping is a transitive relation, therefore the following statement holds.

Statement 2 *Given a sentence with chunks i, j and k, if $(i > j)\ \wedge\ (j > k)$ then $(i > k)$.*

As a result the relation $(2 > 3)$ is entailed from the two constraints and there is no need to explicitly mention that. Therefore we can take advantage of Statement 1 to obtain a more concise representation:

13. $SI :\ (2 > 1 > 3)$

4.2. Coreference/anaphoric relations

The coreference relation is represented as $(i := j)$ and states that both chunks corefer to the same entity.

14. *Take [1/ "test.txt"] and print [2/ every line] in [3/ the file] .*
$(1 := 3\ ;\ 3 > 2)$

Or euiqvalenly (following Statement 1):

15. $SI :\ (1 := 3 > 2)$

Here is a sentence with an anaphoric relation.[3]

16. *For [1/ every line] that has [2/ a punctuation] at [3/ the end], delete [4/ it] .*
$SI :\ (1 > 3 > 2 := 4)$

":=" is a transitive relation, therefore:

Statement 3 *Given a sentence with chunks i, j and k, if $(i := j)\ \wedge\ (j := k)$ then $(i := k)$.*

It may be controversial whether coreference relation should be annotated as a scope relation or not. We label it for the following reasons:

[2] Throughout this paper, unless otherwise specified, the scopings given for every sentence are for the most preferred reading(s).

[3] Throughout the rest of this paper, whenever we talk about coreference relations we are including anaphoric relations as well.

- In sentences with co-referring NPs, specifying coreference relations before trying to figure out scope interactions was shown to lower the chance of error.

- For the very same reason, scoping is more readable to the users of the corpus.

- Since identifying coreference relations helps human to find scope interactions, it may in fact increase the precision of automatic scope disambiguation too.

- Last but not least, it sometimes prevents theoretical problems. For example, in sentence 14 it may be controversial whether chunk 1 directly interacts with chunk 2 or its interaction is only through chunk 3. By explicitly stating the coreference relation between 1 and 3, we have a representation which is both intuitive and theoretically sound. This is particularly important in sentences with donkey anaphora as in 16.

4.3. Bridging anaphora

From quantifier scoping point of view, bridging anaphora are very different from the coreference relation in that they are often a special case of the outscoping relation. Of course this could only be true if the antecedent is accessible to the anaphoric expression as demonstrated in 17.

17. *For [1/ every line] starting with [2/ a question mark] , delete [3/ the last two words] .*
$SI:\ (1 > 2\,;1 > 3)$

In the most preferred reading of this sentence, 1 outscopes 3. At the same time, there is a bridging anaphora between the two chunks. In these cases there is no advantage in annotating bridging anaphora. However, when the antecedent is not accessible to the anaphoric expression, such as in donkey sentences, it is not theoretically sound to consider a scope interaction between the two entities, therefore only in these cases we label bridging anaphora relations.

18. *For [1/ each word] which contains [2/ a hyphen], print [3/ the surrounding letters] .*

In a reading of 18 in which 3 refers to the "surrounding letters" of "a hyphen", informally speaking, 2 has wide scope over 3; but since 2 is not accessible to 3, we cannot have $(2 > 3)$. In these cases we use the bridging anaphora relation represented as "$=>$" to express the scope interaction between the two entities. This symbol emphasizes that bridging anaphora are a special case of an anaphoric relation in which the pronoun referring to the antecedent is absent. For example, in 18, we can imagine that there is an implicit pronoun "its" in chunk 3 (i.e. "[3/ [4/ its] surrounding letters]"), not realized in the surface of the sentence. Therefore $(2 => 3)$ is in fact a short for $(2 := 4 > 3)$. Following this, we represent the scoping of 18 as in 19.

19. $SI:\ (1 > 2 => 3)$

Note that we do *not* consider "$=>$" as a transitive relation.

4.4. Equivalence interaction

Consider the following sentence:

20. *For [1/ each line] add [2/ a colon] at [3/ the end] of [4/ the first word].*

What is the scoping between 3 and 4 in 20? Our experiments show that since "the end" semantically depends on "the word", most people believe $(4 > 3)$, even though, from logical perspective, both scopings are equivalent (partly because both chunks are definite):

21. $\mathscr{D}x\ \mathscr{D}y,\ end(x) \wedge word(y) \wedge of(x,y)$

$\mathscr{D}$ is the generalized quantifier representing a definite NP. Although it is theoretically sound to follow annotators' intuition (since the two scopings are equivalent), it creates a problem in practice. In order to explain the problem, let's consider chunks 2 and 3 in 22.

22. *Delete [1/ every line] containing [2/ a word] wrapped within [3/ single quotes]*

2 and 3 are both existentials and their two scopings are logically equivalent. However, this time different annotators have different intuitions. Some prefer $(2 > 3)$ and some $(3 > 2)$. Therefore relying on annotators' intuition creates inter-annotator disagreement even though both scopings results in the same interpretation. We have required annotators to recognize pairs with semantically equivalent scopings and label them with a new relation, represented by *comma* and called *equivalence interaction*.

23. $SI:\ (1 > 2, 3)$

The relation "," has the following properties.

Statement 4 *Given a sentence with the chunks* $i, j, k, \ldots$

- *if* $(i > j, k, \ldots)$ *then* $(i > j\,; i > k\,; \ldots)$
- *if* $(j, k, \ldots > i)$ *then* $(j > i\,; k > i\,; \ldots)$

We cannot automatically annotate the equivalence relations for several reasons. First, because our corpus is not labeled with the true quantification of NPs (e.g. a bare plural may be a definite expression, an existential, etc.). Second, we avoid making any speculation about the semantics of natural language, so we do not take for granted that every two identical quantifiers have semantically equivalent scopings.[4] Finally it is not the case that only identical quantifiers can have equivalent scopings. There are cases where different scoping of two non-identical quantifiers does not create different interpretations, as in the following example.

24. *For [1/ every line], delete [2/ the last word] starting with [3/ an uppercase letter]*
$SI:\ (1 > 2, 3)$

This is almost always the case for definite NPs vs. existentials in our corpus, but once again following our no-speculation rule, we leave it to the annotators' judgement to decide whether the scopings are equivalent or not.

[4] In fact some quantifiers, such as "no" (because of the implicit negation), are not interchangeable. Although we haven't found such examples in our corpus, they can happen in our domain, as in "*Delete all lines containing no word with no uppercase letter.*" (many thanks to Lenhart Schubert for motivating this example).

4.5. Scoping with scopal operators

Trivially, there is no coreference or bridging anaphora relations for scopal operators. Therefore the only relations between two chunks where one of them is a scopal operator is either an outscoping relation or no interaction. Sentence 8 (Section 3) is repeated below with two plausible scopings.

25. *Cut [1/ every word] that occurs [O1/ twice] in [2/ a line] .*
    (a) $SI: (1t > 2 > O1)$ */*Each word in the text will be removed, if there exists a line containing two occurrences of that word.*/*
    (b) $SI: (2 > 1t > O1)$ */*Given a line, for every word that occurs twice in the line, both instances of that word will be removed from the line.*/*

In the above scopings, $1t$ is used to refer to the first chunk. You can ignore the suffix "t" for the moment. In Section 6, we describe what the letter "t" stands for.
Example 26 shows scoping of a sentence with negation.

26. *Erase [1/ every line] that does [N1/ not] contain [2/ a digit].*

    $SI: (1 > N1 > 2)$ */*A word is removed if none of its characters are numeric i.e. [0-9].*/*

There is another plausible scoping for this sentence with "a digit" outscoping the negation. It corresponds to the reading in which the "line" is removed if there exists "a digit" such that this "digit" does not exist on "the line". We haven't yet defined the mechanism to represent this reading (see 45 in Section 6).
The scoping of sentential operators is slightly different. Those operators often create two or more subspaces, e.g. the antecedent and the consequent for conditionals. We represent those subspaces by $i.1, i.2, \ldots$ for a sentential operator with ID i. 27 gives an example of a sentential operator.

27. *[S1/ If] there is [1/ a line] ending in [2/ a digit] , delete [3/ every alphanumeric character] from [4/ it] .*
    $SI: (S1.1 > 1, 2\ ;\ S1.2 > 4 > 3\ ;\ 1 := 4)$ */*1 and 2 are both considered to be existentials.*/*

Subspaces of a sentential operator are considered orthogonal, that is the entities (scoped) within one subspace are not accessible to the entities within another. For example, there is no interaction between an entity within the scope of the antecedent and one within the scope of the consequent of a conditional, except through a coreference or bridging anaphora relation.[5]

4.6. No interaction

Given two chunks i, j, if no outscoping, coreference, or bridging anaphora relations between the two are *explicitly mentioned* in or can be *entailed* from the given scoping, then the two chunks are considered to have *no interaction*.
For example, chunks 2 and 3 in 17 have no interaction. There are several types of NP which never have any scope interaction with any chunk in the sentence. Those chunks are explicitly labeled in the scoping. The next two subsections describes two types of NP with this property.

[5]In Discourse Representation Theory (Kamp et al., 2011) the antecedent is considered accessible to the consequent, but our annotation scheme is not designed for a particular semantic formalism, hence we do not consider such an accessibility.

4.6.1. Constants

Trivially for any given domain, the *constants* of the domain do not participate in scoping. The suffix ":C" is attached to a chunk representing that the chunk denotes a constant. Some example of constants in our domain are constant integers such as 1, 2, 3, etc., constant characters such as "1", "2", "a", "b", "(", ")", etc., and constant strings such as "dog", "cat", "mouse", etc.[6]

28. *Replace [1/ every occurrence] of [2/ dog] with [3/ an occurrence] of [4/ cat] .*
    $SI: (2 : C\ ;\ 4 : C\ ;\ 1 > 3)$

4.6.2. Non-scopal adverbials

As mentioned before, we chunk every NP in the sentence, leaving it to the annotator to decide whether the NP participates in the scoping. An example of NPs with no interaction are NPs acting as an adverbial (often as part of a PP). For example consider the following sentence:

29. *For [1/ every file] in [2/ the folder] sort [3/ the lines] alphabetically ignoring [4/ case].*

One can argue that since "the lines" are composed of "characters" and each "alphabetical character" has a "case", chunk 4 is in fact participating in scoping. Another interpretation is that "ignoring case" is simply an adverbial modifying the verb "sort" similar to the adverbial "alphabetically"; therefore, "case" does not need to be scoped. The first approach could be adopted if we had a detailed semantic representation of the sentence which supported this interpretation. Since we do not have such a detailed semantic representation, we would prefer to go with the second interpretation. A suffix ":P" (for predicate) is attached to the chunk's ID in such cases to indicate that the NP chunk behaves as an adverbial and does not introduce a first order entity that participates in scoping, Following this interpretation, here is a scoping for the sentence in 29.

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

4.7. No interaction vs equivalence interaction

It is implicit in the definition of *no interaction*, at the beginning of Section 4.6, that equivalence interaction between two chunks is treated like no interaction. While some work in the past (Higgins and Sadock, 2003) has adopted the same strategy (that is, no distinction between equivalence interaction and no interaction), some make a distinction between the two (Galen and MacCartney, 2004). We follow the first strategy for the following reasons.
First, the boundary between the two cases is very blurry. Consider the following sentence.

31. *Delete [1/ the first letter] of [2/ the first word] and [3/ the last letter] of [4/ the last word] in [5/ each row] .*

Clearly 5 has wide scope over 1 & 2 ($5 > 1, 2$) and 3 & 4 ($5 > 3, 4$). However, it is not easy to decide whether there is a scope interaction between 1 and 3. Arguments can be made in favor of either alternative. In fact, our experiments show that neither of the alternatives is highly preferred to the other by the annotators. This is not surprising,

[6]These are constants as a type not as a token (cf. Section 6).

as it makes no difference in practice. This last point forms our second argument in favor of "no distinction" strategy. Whether 1 and 3 have no interaction or they do interact but both scopings are semantically equivalent makes no difference in how one interprets the overall meaning of the sentence or perceives the requested task. This suggests no distinction between the two relations, resulting in the three following scopings for 31 being considered equivalent.

32. **(a)** $SI:\ (5>1\ ;\ 5>2\ ;\ 5>3\ ;\ 5>4)$
(b) $SI:\ (5>1,2\ ;\ 5>3,4)$
(c) $SI:\ (5>1,2,3,4)$

Note that although it may show counter-intuitive behaviors, our notation potentially supports the distinction. To see why, remember that from statement 4, $(5>1,2)$ entails $(5>1\ ;\ 5>2)$. If we distinguish the two relations, the other direction does not hold; because the former states that 1 and 2 have equivalence interaction, while the latter implies that there is no interaction between 1 and 2. Therefore, the three scopings in 32 will no longer be equivalent.[7]

5. Dealing with plurals

Plurality and quantification have been one of the most challenging phenomena in theoretical semantics (Hamm and Hinrichs, 2010). Although the problem has been studied in depth in theoretical semantics, the focus has been mainly on the interaction of events and plurality (Landmann, 2000). A typical example of this phenomenon is given in 33.

33. *The students met the teacher.*

In this sentence, the ambiguity is whether there is a single event (*collective* reading) or multiple events of meeting (*distributive* reading). The complexity could go beyond the simple two possible readings in general, as in the following example from Gillon (1987).

34. *Rodgers, Hammerstein, and Hart wrote musicals.*

where any of the the three musicians did not write a musical all by himself nor all three collaborated to write any musical. Fortunately, these cases are rare and at least in our corpus we do not encounter such complex cases. However, we have to deal with an issue which often has not been addressed in the literature. It is not enough for us to decide the type of the reading carried by a plural per sentence (which is the common practice in theoretical semantics), but we have to make such a decision for every pair of scope-bearing elements in the sentence (where at least one is a plural). This issue may or may not be interesting from a theoretical perspective, but it is definitely a problem from computational point of view.
To further support our claim, consider the following sentence in our domain:

35. *Find [1/ the count] of [2/ rows] containing [3/ a numerical field].*

Relative to chunk 1, the plural must be treated as a *collection* (a.k.a *group*), hence has a collective reading, but with respect to chunk 3 it most likely has a distributive reading (each row has its own (potentially distinct) numerical field). A common practice to describe the collectivity vs distributivity is to treat plurals as a *collection* of *individuals*. We adopt the same strategy but we frame it into a more general realm of *terms introducing more than one entity in the domain of discourse*, which we have used to describe some other phenomena as well (cf. Section 6). Therefore chunk 2 introduces two entities: a collection and the individuals in the collection. We represent the collection as $2c$ and the *universally quantified* entity, ranging over the individuals in the collection, by $2d$. This means that we will be scoping four instead of three entities.

36. $SI:\ (1, 2c>2d>3)$

The relation $(1, 2c)$ is the result of the fact that chunk 1 and chunk 2 (as a collection) are both definites. $(1>2d)$ emphasizes that with respect to chunk 1, the plural has a collective reading. That is, there is a definite "number", the size of the collection, which is unique for all the individuals in the collection. In contrast, the plural has a distributive reading with respect to chunk 3, because (in the most preferred reading) each individual has its own potentially distinct numerical field. Therefore, representing a universal quantifier, $2d$ outscopes the existential in chunk 3.
Although less frequently, more complex sentences with plurals occur in our corpus, as in the following sentence.

37. *Print [1/ the sum] of [2/ numerical fields] in [3/ all the rows].*

37 demonstrates a true collective-vs-distributive ambiguity (that is there are in fact two plausible readings). The simple machinery devised above can handle these cases too and represent both plausible readings, as shown in 38 and 39.

38. $SI:\ (1, 2c, 3c>2d, 3d)$ */*The reading in which a single sum is printed.*/*

39. $SI:\ (3c>3d>1, 2c>2d)$ */*The reading in which a distinct sum is printed for each individual row.*/*

To better understand 38 and 39, we have represented their corresponding logical formulas in 40 and 41 respectively.

40. $\mathscr{D}x_1\ \ \mathscr{D}x_{2c}\ \ \mathscr{D}x_{3c}\ \ N(x_1)\ \wedge\ C(x_{2c})\ \wedge\ C(x_{3c})\ \wedge$
$[\forall x_{2d}\ \ \forall x_{3d}\ \ (F(x_{2d})\ \wedge\ R(x_{3d})\ \wedge\ In(x_{3d}, x_{3c})\ \wedge$
$In(x_{2d}, x_{3d}))\iff In(x_{2d}, x_{2c})\,]$
$\wedge\, S(x_{2c}, x_1)\wedge P(x_1)$

41. $\mathscr{D}x_{3c}\ C(x_{3c})\ \wedge\ \forall x_{3d}\ [\ (R(x_{3d})\ \wedge\ In(x_{3d}, x_{3c}))\ \iff$
$(\ \mathscr{D}x_1\ \mathscr{D}x_{2c}\ N(x_1)\wedge C(x_{2c})\wedge$
$[\,\forall x_{2d}\,(F(x_{2d})\wedge In(x_{2d}, x_{3d}))\iff In(x_{2d}, x_{2c})\,]$
$\wedge\, S(x_{2c}, x_1)\wedge P(x_1)\ )\ ]$

[7] There are compelling arguments in favor of the distinction as well, even when we look at the issue solely from practical point of view. For example, two universals with no interaction could represent two independent loops, while two universals with equivalence interaction may be translated into a nested loop structure. Further investigation of this matter is left for future work.

Where N, C, F, R, S, and P stand for *Number, Collection, Field, Row, Sum,* and *Print* respectively.[8]
By convention, chunk ID i with no suffix is equivalent to ic. This helps to have a regular scoping for sentences in which the plural has no distributive reading, as in 23 for 22.

6. The type-token distinction

Consider 42, purposefully chosen from outside our domain.

42. *[1/ Every student] in [2/ the class] had [3/ the text book].*
$SI: (2, 3 > 1)$ */*Incomplete scoping*/*

Even though the scoping is given, the sentence is still ambiguous between two readings, one where there is a single copy of the textbook and one where there are multiple copies. This happens because it is not clear what 3 in the given scoping is referring to, "the textbook" as an abstract entity, say Landmann (2000), or "the textbook" as a concrete physical instance, say the copy of Landmann (2000) in room 631 of the Computer Studies Building at the University of Rochester at the time of writing this paper. "the textbook" in its first usage is called a *type* and in its second usage is referred to as a *token* (Peirce, 1958). Note that this is not a word sense ambiguity problem because both senses are present at the same time. The phenomena is called *type-token distinction* and has been well studied in philosophy and formal semantics. In computational semantics, however, it has not been investigated in depth. In particular, to the best of our knowledge, there is no previous work on dealing with type-token distinction in the realm of quantifier scope disambiguation.
Incidentally, the type-token distinction is a big (if not the biggest) issue in our domain. It was not surprising for us to learn that the popular example for demonstrating this phenomenon happens to be in our domain.[9] It addresses the usage of "word" as a type (as in "the word dog" in its abstract sense) vs. its usage as a token (say the physical realization of "dog" in your copy of this paper). In fact almost all the examples we have given in this paper so far deal with this issue. This is because most concepts in our domain ("Line", "Word", "Character", etc.) may be perceived as an abstract entity (type) as well as a physical realization (token) of this entity. As an example, consider the following sentence.

43. *Delete [1/ every line] that ends with [2/ a digit] .*
$SI: (1 > 2)$ */*Incomplete scoping*/*

In the above scoping, chunk 2 may refer to "a digit" as a type or a token. In the latter case, the scoping states the obvious: that "every line" has its own instance of "a digit", so it is still ambiguous whether the type of "digit" is the same for "every line" or not.

[8] As seen in 39 and 41, we are able to represent the distributive reading without reification of the predicate *print*. That has generally been the case in our corpus. For other domains, one may need to use a neo-Dvidsonaian approach to represent events in order to be able to represent both distributive and collective readings of a plural with respect to an event.

[9] Consider the line "Rose is a rose is a rose is a rose." from the poem *Sacred Emily* by Gertrude Stein, borrowed from Wetzel (2009). How many words are in this line?

Here we use the same mechanism, which we already devised to deal with plurals. That is we treat the type-token terms as another category of nouns introducing more than one entities in the domain of discourse. One of the entities is the type and the other is the token. We use the suffix "t" to represent the type and the suffix "i" (for "instance" or "inscription") to represent the token.

44. $SI: (1 > 2t, 2i)$

In 44, $2t$ refers to "a digit" as a type, contrasting $2i$ which refers to "a digit" as a token. This scoping corresponds to the reading in which "every line" ends with a potentially distinct type of "digit". Note that while both type and token for "a digit" are within the scope of universal, they are both existentially quantified, hence carry equivalence interaction, resulting in the scoping in 44.
As another example, consider the sentence in 9, repeated below with its scoping revised using the new machinery, in order to be able to assign a second scoping to the sentence.

45. *Erase [1/ every line] that does [N1/ not] contain [2/ a digit].*
(a) $SI: (1 > N1 > 2i, 2t)$ */*A line is removed if none of its characters are digit.*/*
(b) $SI: (1 > 2t > N1 > 2i)$ */*A line is removed if there exists a type of digit that does not appear on this line.*/*

In the preferred reading of almost all the examples in this paper, the token in the type/token chunks outscopes the type (that is every token is of a potentially distinct type, as in "every line in the file", "every word in a line", etc.), therefore we choose this scoping as the default for type/token chunks. Furthermore, for a type/token chunk m, we assume that m (with no suffix) is equivalent to mi, the index representing the token entity. Using these two conventions, in most cases we do not have to worry about the type-token distinction and only scope the *token* entity introduced by chunks.[10] As a result, almost all scopings given in the previous sections remain correct with no modification.[11]

7. Modeling scoping as a DAG

Although we have coreference (and bridging anaphora) relations annotated in the corpus for the reasons we have discussed above, our true goal is to model scoping. Annotating coreference relations is by far easier than scope disambiguation, and incorporating those relations in the evaluation metric may result in deceptively higher numbers when measuring IAA or the performance of an automatic scope

[10] 25 is the only exception in which the token interpretation of "word" is implicit in the verb "occurs", therefore chunk 1 does not make a direct refernce to a token and only introduces a type.

[11] The type-token problem is more complex in general. In fact, it is not enough to consider one layer of abstraction. Consider the sentence "Insert [1/ a line] containing [2/ 10 consecutive '*'] after [3/ every other line]". Chunk 2 represents neither a type (otherwise, what are we counting "10" of?), nor a token (because chunk 1 is intrdoucing an abstract entity (a type of line), hence it cannot "contain" a token (i.e. a phyisical realization)). To explain chunk 2, we need to define a second layer of abstraction, called *occurrence* (see Wetzel (2009)). We cannot afford to take this into account at the moment, as the scheme is already quite overwhelming for the annotators. We leave that to the future.

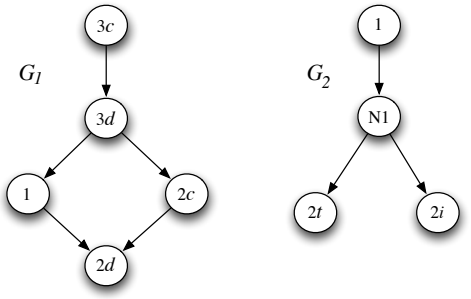

Figure 1: Scoping as DAG

disambiguation system. Therefore we exclude them from our model of scoping in order to define evaluation metrics. Excluding those relations and given that we make no distinction between no interaction and equivalence interaction relations leaves us with only two types of relation: outscoping or no scope preference (where the latter includes both no scope interaction and equivalence interaction). We can simply model those two relations using a graphical notation in which the scope-bearing elements represent the nodes and outscoping relations form the directed edges. Therefore if two nodes are not connected by an edge, there is no scope preference between them. Since outscoping constraints cannot form a cycle in a well-formed scoping, the graph will be a *directed acyclic graph* (DAG). G_1 and G_2 in Figure 1 are the DAGs for the scopings in 39 and 45(a) respectively. In order to define evaluation metrics, we need to define a similarity measure between two DAGs, which can be used in calculating both IAA and the performance of an automatic scope disambiguation system against gold standard data. Given two DAGs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ the similarity measure is defined as:

$$\sigma^+ = \frac{|E_1^+ \cap E_2^+| + |\bar{E_1^+} \cap \bar{E_2^+}|}{|V|(|V|+1)/2} \quad (1)$$

in which $\bar{G} = (V, \bar{E})$ and $G^+ = (V, E^+)$ are the *complement of the underlying undirected graph* and the *transitive closure* of the DAG $G = (V, E)$ respectively. Details can be found in Manshadi et al. (2011). There, we also define precision/recall for evaluating the performance of an automatic scope disambiguation system vs. gold standard data. We have achieved the IAA of 75% on our corpus, much higher than 52% agreement that Higgins and Sadock (2003) achieve for a simpler task but broader domain.

8. Conclusion

We present the first work on developing an annotation scheme for quantifier scope disambiguation when there are an arbitrary number of scope-bearing elements (including all NPs and most scopal operators) in the sentence. Our scheme presents machinery for dealing with some complex scope phenomena including plurality and the type-token distinction. Some details are left out because of the space limitation. In particular, we define the scheme in an informal way with the help of intuitive examples, which we hope will suffice to understand the scope annotations in our corpus. Besides, there are complexities that have not yet been addressed in the current scheme. A formal definition of the framework and solutions to some of the issues that the current scheme is dealing with remains for the future.

9. Acknowledgements

We need to thank Eric Meinhardt and Timothy Dozat for their feedbacks on the annotation scheme, and Lenhart Schubert and Greg Carlson for the fruitful discussions. This work was supported in part by NSF grants IIS-1012205 and IIS-1012017, and ONR grant 00014-11-1-0417.

10. References

J. Allen, M. Dzikovska, M. Manshadi, and M. Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of EACL-07 Workshop on Deep Linguistic Processing*.

J. Bos. 2008. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP '08, pages 277–286.

A. Copestake, D. Flickinger, C. Pollard, and I. A. Sag. 2005. Minimal recursion semantics – an introduction. *Research on Language and Computation*, 3:281–332.

A. Galen and B. MacCartney. 2004. Statistical resolution of scope ambiguity in natural language.

B. S. Gillon. 1987. The readings of plural noun phrases in english. *Linguistics and Philosophy*, 10:199–219.

F. Hamm and E. W. Hinrichs. 2010. *Plurality and Quantification*. Studies in Linguistics and Philosophy. Springer.

D. Higgins and J. M. Sadock. 2003. A machine learning approach to modeling scope preferences. *Comput. Linguist.*, 29(1):73–96, March.

H. Kamp, J. Genabith, and U. Reyle. 2011. Discourse representation theory. In Dov M. Gabbay and Franz Guenthner, editors, *Handbook of Philosophical Logic*, volume 15, pages 125–394. Springer Netherlands.

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

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*.

M. Manshadi, J. Allen, and M. Swift. 2011. A corpus of scope-disambiguated english text. In *Proceedings of ACL-HLT '11: short papers*, pages 141–146.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The lingo redwoods treebank motivation and preliminary applications. In *Proceedings of COLING '02*, pages 1–5.

C. S. Peirce. 1958. *Collected Papers of Charles Sanders Peirce: Science and Philosophy and Reviews, Correspondence and Bibliography*. Harvard University Press.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040.

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

L. Wetzel. 2009. *Types and tokens: on abstract objects*. MIT Press.