Referring Expressions with Groups as Landmarks

Stefan Weijers
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
s.weijers@student.utwente.nl

ABSTRACT

In this paper an algorithm to produce referring expressions in complex scenes is described, evaluated and compared to the Sequence of Groups algorithm [5]. It shows that the new algorithm using groups as landmarks produces expressions that are easier to understand for the hearer. It discusses methods humans use to produce referring expressions in these type of context sets and what improvements could be made to the algorithm described.

Keywords

Referring Expressions, Groups, Locative Incremental Algorithm, Sequence of Groups

1. INTRODUCTION

Referring expressions are an important part of any natural language generation (NLG) system. A referring expression in the context of this research is a description of an entity that enables a hearer to uniquely identify the target in a given visual context. For example "the red chair in the corner". All the entities in the context are called the context set.

Current referring expression generation algorithms can identify almost anything in diverse scenes. This does not always mean that the expressions generated are understandable for human hearers. A situation that is considered hard to generate expressions for is when the context set contains many entities with the same properties. Almost all current algorithms will fail when the context set contains only one type of entity.

Kelleher and Kruijff [4] proposed a method to identify non-unique entities. When the target entity is not distinguishable they use a different entity that is easily identifiable to guide the hearer to the target.

Those kind of entities are called candidate *landmarks*. The algorithm then tries to construct a referring expression that is in some spatial relation to the landmark. (i.e. to the right, on top of, in front of) Landmarks with higher salience are preferred above landmarks with low salience, therefore larger or more unique landmarks are preferred above landmarks that are not so visually present.

15th Twente Student Conference on IT June 20st, 2011, Enschede, The Netherlands.

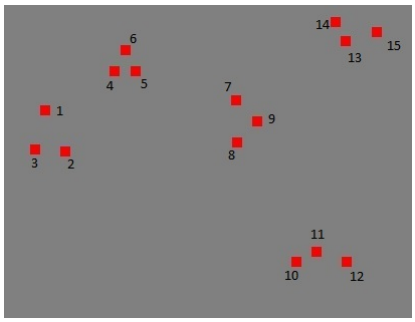

Figure 1. A situation the SOG algorithm [5] handles sub-optimal.

This algorithm cannot produce a referring expression if the context set only contains entities with the same properties (therefore no landmarks). An algorithm that can refer to objects even if there are no landmarks is the Sequence of Groups (SoG) algorithm by Kotaro, Satoru and Takenobu [5]. This algorithm splits the complete context set into different groups based on their properties (which have discriminatory power). For all the groups that are split in the former step, this process is repeated (so split again with all possible properties) till the set only contains a single entity. If this entity is the target, the properties needed for a referring expression are found. If it is not the target this single entity is tested for spatial relations with the target. Therefore the SoG algorithm can recursively go through all entities and relations till it finds a path that singles out the target.

If I wanted to refer to square 9 in figure 1 without using the numbers using the SoG algorithm, I would get something like: " The rightmost square left of the leftmost square in the bottom-most group of three".

$\{All\} \overset{bottom}{\rightarrow} \{10,11,12\} \overset{left}{\rightarrow} \{10\} \overset{left}{\rightarrow} \{9\}$

$\{All\} \overset{top}{\rightarrow} \{13,14,15\} \overset{left}{\rightarrow} \{14\} \overset{left}{\rightarrow} \{11\} \overset{left}{\rightarrow} \{10\} \overset{left}{\rightarrow} \{9\}$

$\{All\} \overset{left}{\rightarrow} \{1,2,3\} \overset{right}{\rightarrow} \{2\} \overset{right}{\rightarrow} \{4\} \ldots. \{7,8\} \overset{right}{\rightarrow} \{9\}$

The shortest version would be the first one. Resulting in the expression mentioned above.

A disadvantage of the strategy used in the SoG algorithm is that it needs to refer to a single entity before relating it to a new target. It can be argued that groups have a higher salience than a single entity, therefore it would seem to be better to use an easily distinguishable group with high salience to refer to, rather than some hardly distinguishable entity within such a group. Hopefully referring expressions using groups as landmarks can create

more logical referring expressions that humans can easily understand.

In the figure 1 example, a more desirable expression would be: "The rightmost square in the group of three, that is most right to the left of the bottom-most group of three". The algorithms described in this paper rely heavily on group detection algorithms. For example the SoG uses Thórissons algorithm [7] to detect visual groups and use them in the expressions.

Before we can compare algorithms we should know what measure to use. The domain is quite complex and it is easy for the human hearer to make mistakes when identifying an object. The main focus on our expressions will be on the understandability of the expressions generated. We define understandability as the accuracy and speed of identification, as done in the Tuna '08 experiments [3]. For this we assume the hearer has no problem with the language used and that the properties used by the expression are "easy to see" for the hearer.

By using the above definition of "understandable" the main research question would condense to: *How can we make an algorithm that can produce referring expressions with groups as landmarks that are understandable for humans?* To be able to answer this question a number of sub-research questions are needed.

- To what extent are the generated referring expressions understandable?
- Are referring expressions with groups as landmarks preferred over referring expressions with a single object as a landmark? If so, to what extend?
- In what context-sets are referring expressions with groups as landmarks suitable? And how can these situations be identified?

To answer these sub questions we will perform an evaluation of the expressions generated by the algorithm, and compare these to a simulated version of the SoG algorithm. Section 2 will explain the group based algorithm in detail. Section 3 is about the evaluation and the results. Section 4 discusses the findings. The final section will summarize briefly the findings and answers the main research question.

2. ALGORITHM

A part of the complexity in the expressions generated by the SoG algorithm [5] in these kind of context sets, is that it does not use groups as landmarks to jump to another group. The SoG algorithm just uses single entities. Another algorithm that does this is the locative algorithm [4].

The locative algorithm uses some entity it can identify and uses spatial relations with the other entities to refer to the actual target. In context-sets containing a lot of entities with (nearly) the same properties we would like to refer to groups instead of single entities, because referring to many entities one by one would result in a quite long expression.

The Group-Based Locative algorithm (GBLA) proposed in this paper is an extension of the Locative algorithm, so that it can handle groups. The GBL algorithm has two sub algorithms that are used to generate the final expression. Algorithm 1 calls the main algorithm and calls the function to identify the groups in the context set. An algorithm that can identify these groups would be a modified version of Thórissons algorithm [7]. The modification boils down to making sure that the groups are mutually exclusive, as this algorithm asumes all entities are at most in one group. In section 3.2.1 there is a situation where this assumption was not true for all hearers.

Algorithm 1 uses the groups in two ways. First it identifies the target entity in the group it is in using the Incremental algorithm.

Algorithm 1 Start-up algorithm

Require: CS = Context-set, E = target entity
  DetermineGroups(CS)
  E ∈ G
  DESC = IncrementalAlgorithm(G, E)
  DESC +=GroupBasedLocativeAlgorithm(CS,G)
  return DESC

The Incremental Algorithm (IA) [1] identifies its target by selecting a set of properties that distinguish it from all others (distractors). The properties are sorted and considered by the IA one by one. If the current property excludes any entity in the distractor set, that property is added to the list that is returned and the distractor set is reduced. When the distractor set is empty return the list of properties that contributed. This list always contains at least the type of the object. The IA fails if any of the entities has exactly the same properties as the target entity.

In contrast to the IA described by Dale and Reiter [1] this version only considers entities in its own group as distractors. Therefore the resulting expression should be relatively short. The IA used for the evaluation in section 3 used a coordinate system to identify an entity within a group, because it is required to return a useful expression. If the group that contains the entity is still too big, it could be considered to run the complete GBLA (including the start-up algorithm and thus the group-detection algorithm) again with that group as complete context set. That way it could determine subgroups within that group and reduce the size of the expression even more. Small groups (less than 6 entities) can easily be handled by a coordinate system (see section 4.1 for details).

After determining the in-group referring expression it turns to generating the expression for the group itself. Here the GBLA (algorithm 3) is used. The way this algorithm works is explained further below. When the Group-Based Locative algorithm completed execution, the return values are the properties and relations needed to identify the group that contains the entity.

Before the workings of the Group-Based Locative algorithm (algorithm 3) can be explained, the Basic Group Incremental Algorithm (BGIA) needs to be discussed (algorithm 2). This is practically the same thing as the normal Incremental Algorithm (IA) [1]. The only real difference is that it tries to uniquely identify a group instead of a single entity. It uses properties of the group such as the number of elements, the location, the colour of its elements or its shape, to come up with a unique expression of this one group. As with the normal IA [1], the BGIA can not always produce a distinctive description. When the group does not have any discriminatory feature that excludes it from all other groups, the interesting situations that require the Group Based Locative algorithm occur.

The Group-Based Locative algorithm (GBLA) is an adaptation of the Locative Incremental Algorithm (LIA) [4]. Like the LIA the GBLA uses other items as landmarks and checks if they can be used in a relation with the target (group). In essence the GBLA only differs from the

Algorithm 2 Basic Group Incremental algorithm

Require: G = target group; D = set of distracter groups.
P ={type, colour, size, location}; DESC = {}
for i = 0 to |P| **do**
if $G_{salience()}$ > MAXDISTRACTORSALIENCE **then**
Distinguishing description generated
if $type(x) \notin$ DESC **then**
DESC = DESC ∪ type(x)
end if
return DESC
else
$D' = \{x : x \in D, P_i(x) = P_i(G)\}$
if $|D'| < |D|$ **then**
DESC = DESC ∪$P_i(G)$
end if
$D = D'$
end if
end for
Failed to generate distinguishing description
return DESC

Algorithm 3 Group-Based Locative algorithm

Require CS = Context-set, G = Group Containing the Target
DESC = *BasicGroupIncrementalAlgorithm*(CS, G)
if DESC not distinguishing **then**
CL is set of Candidate landmark groups
MAP can store a Group and Relation.
for all Group TL∈ CL sorted on salience **do**
for all Relation R in relations sorted by relevance **do**
if Relation Exists ($TL \rightarrow G$) **then**
TLDESC = *BasicGroupIncrementalAlgorithm*(CS, TL)
store R, TL in MAP
if TLDESC is Distinguishing **then**
CSR = ReduceContextSet(R)
DESC = *BasicGroupIncrementalAlgorithm*(CSR, G)
if DESC is distinguishing **then**
return (DESC ,R, TLDESC)
end if
end if
end if
end for
end for
for all Group TL, Relation R ∈ MAP sorted on salience **do**
CSR = ReduceContextSet(R)
DESC = *GroupBasedLocativeAlgorithm*(CSR, G);
if DESC is distinguishing **then**
return (DESC ,R, TLDESC)
end if
end for
end if

LIA in two points. The GBLA uses groups (obviously) and the GBLA can do recursion to generate expressions where one relation is not enough.

The GBLA works as follows. Firstly the BGIA is called for the target group. If this already results in a distinguishing expression, there is no need for more complexr expression. When the target group in itself is not distinguishable, the list of candidate landmarks is made. This list contains all groups in the context set, excluding the target group itself. Unlike LIA, the candidate list does contain groups that have the same properties as the target group. This might seem useless, since these can never function as a landmark, but to exclude them, the algorithm still needs to do the same checks it does now, thefore it is not more efficient. Furthermore, that group could be used later by using recursion, since that group could be in a relation with a group that is identifiable. We also need some memory to store the groups and relations that are viable for later recursion.

Then, for all the groups in the candidate landmark list and for all relation $\Re$ (think of: right, left , above, below),$\Re$ is tested for the candidate landmark and the target. If $\Re$ is not a valid relation for the target group and the landmark, we continue to the next relation. Note that every group has some relation to the target group, so there are at least two entries in the map for every group. Since there are so many possibilities for expressions and we do not backtrack, we would like to get the best and most obvious one first. Therefore we sort the candidate landmarks on salience[1] and the relation to preference order[2].

If $\Re$ is a valid relation the description for this landmark is determined by the BGIA and $\Re$ stored in the map. If the landmark is distinguishable we reduce the context set to include only the groups that have relation $\Re$ with the landmark. If the group containing the target is now distinguishable using the BGIA, these properties,$\Re$ and the properties of the target landmark are returned. If the group containing the target is still not identifiable we continue with the next group as a landmark till all groups have had a turn. If there is still no distinguishing description, this means that there is no distinguishing (detectable) relation for the group containing the target and any other group.

Now for all the relations and landmarks stored in the map, we reduce the context set based on that landmark and relation and recursively call the GBLA again, resulting in an expression containing at least two relations and landmarks. If for any reason this is still not distinguishable we continue till we have used all the relations the group containing the target has. Finally we return null if no expression could be found. Although this can never happen if spatial relations are used, because with spatial relations there is always a “rightmost” or “leftmost” group that can be used. After a relation there is always a new group the “rightmost” or “leftmost” resulting in an strictly smaller set each iteration. Therefore it must always return some relational path that could be used to identify the group containing the target. Although, as explained in section 3.2, the understandability of such long expressions might render them useless.

[1]The salience is currently determined by the number of elements in the group devided by its size.
[2]To order the relations we check which is higher, the horizontal difference or the vertical difference and use the relations regarding that dimension first. Admittedly the order of these relations is not backed by any research at the moment.

For clarity the GBLA algorithm is illustrated using the context in figure 1. Note that three groups can be identified by being closest to an edge. Namely {1,2,3}, {10,11,12} and {13,14,15}. Let's try to identify number 9. The target group {7,8,9} is not among the identifiable groups, therefore the BasicGroupIncrementalAlgorithm fails. The CL in this scene are all the groups for now. The algorithm now takes one of the groups. Let's say it picks {4,5,6} first. The algorithm finds out that the target group is below and right of this group, but it also discovers that this group is not uniquely identifiable and moves on to the next one. Let's say {13,14,15}. For this group the relation below and left are valid, so for these relations it will make a description of this group. {13,14,15} is uniquely identifiable. Now the context-set is reduced to all groups below or left of this set. In either case no other groups are eliminated, and the current context-set are all groups other than {13,14,15}. Is the target group now identifiable? No, it is not it is still not the closest to any edge. Therefore the relations with {13,14,15} are stored and the algorithm continues with the next group. This time it takes {10, 11, 12}. This group also has two relations with the target group, but this time it excludes two groups if the algorithm takes the groups left of {10, 11, 12}. Now the target group has become the rightmost group and therefore identifiable. The solution presented by the GBLA algorithm is: "the group of three the most right, left of the bottommost group of three".

Some considerations:

- The algorithm is very dependent on the performance of the group recognition system. If that system can detect shapes, lines or other perceptive attributes of a group, these could be used to identify such a group. For the evaluation of the algorithm I used none of these kinds of attributes, since I did not have an algorithm to detect them. The only attributes of groups that were used are the number of entities in a group and the location relative to other groups.

- As with the above, a special case is the use of middle. To determine the middle group is not so trivial and would require quite a bit of research in itself. The use of the middle has deliberately been avoided. Some sets in the evaluation could more easily be identified by using the middle. With an algorithm that could detect what groups are considered in the middle, this information could probably be used to get easier understandable expressions.

3. EVALUATION

The algorithm discussed in section 2 produces descriptions for entities in (almost) all scenes imaginable. That does not necessarily mean they are always good enough that human hearers can actually identify the target. In some scenes not every hearer might agree with all others. Also, although the expressions generated are correct, not every hearer will like or prefer these above others. What is needed is a way to measure the performance of these new expressions. In section 1 we defined understandability as the level of accuracy and speed of identification. That is why we will test the expressions generated for these properties. The identification time and accuracy will be measured at the same time, but evaluated separately[3]. With these points of data we can conclude something about the preference of hearers and identify what kind of context sets are problematic or easy.

3.1 Set up

For the evaluation 10 contexts were made of various difficulty. I would like to refer to difficulty as the number of inter-group relations needed for the expression. Degree one would be that there were no inter-group relations needed to identify the group in which the target resides. For example a context set where the rightmost group had the target entity. Obviously, we could refer to that group directly, using "the rightmost group" as property. Degree two would be the groups that would need one inter-group relation, before we reach the group containing the target. Generally these are considered to be more difficult for the hearer. Similarly, difficulty degree three context sets would require two inter-group relations.

In the evaluation only the edges of the scenes are used for locative information (thus, right, left, up, down; and not middle or upper right) for the properties of BGIA and the relations in GBLA, although many other spatial relations could be imagined. Due to time constraints only these four were present for evaluation. Note that having more types of spatial relations decreases the difficulty level of the expressions, which would also mean the scenes have to be made even larger to retain the same degree of difficulty. Also note that due to the nature of the degree of difficulty and the use of the edges as spatial relations, the expression of maximum difficulty for that set refers to a group somewhat in the middle. In other words at least n similar groups away from any side of the scene where $n+1$ is the degree of difficulty.

Every one of the 10 context sets used an expression of about maximum difficulty for that set. The expressions were generated using an implementation of the GBLA in Java. The figures used in this paper were the input for the implementation.

The first context set was used to explain the idea of the survey to the respondents. This expression was only of difficulty level one. The real test consisted of sets 2 to 10. Sets 2 to 7 had expressions of difficulty level two, while sets 8 to 10 had expressions of difficulty level three. For this evaluation the difficulty level was capped at three, since these sets[4] are already quite complex. Even more complex scenes would probably ask too much of the respondents and not result in that much useful data.

Just the experimental information from the expressions generated by the GBLA is not enough to conclude much. The results of the evaluation need to be compared to similar expressions generated by some other algorithm. The SoG [5] algorithm can also generate expressions for the sets used in the evaluation. Furthermore, it uses a quite different strategy, so the resulting expressions are quite different as well. For all the context sets the SoG algorithm was simulated. With these 10 sets and 20 expressions two evaluations were made. The evaluations had about equal number of SoG and Group-based expressions and the expressions were all used. The easiest scene had the same expression for SoG and GBLA, therefore this question has not been used for the actual data gathering.

Although the sets are ordered by intended difficulty, in the survey the questions were in random order, and the two

[3]As in the total performance is not a concatenation of the two, but rather two separate measures that could possibly contradict each other

[4]Although difficulty is determined in the expression, the maximum difficulty of any expression is determined by the set. Therefore, expressions of higher difficulty level require more complex sets.

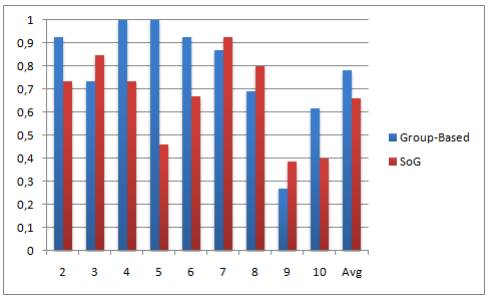

Figure 2. The accuracy measured during the survey. Higher is better.

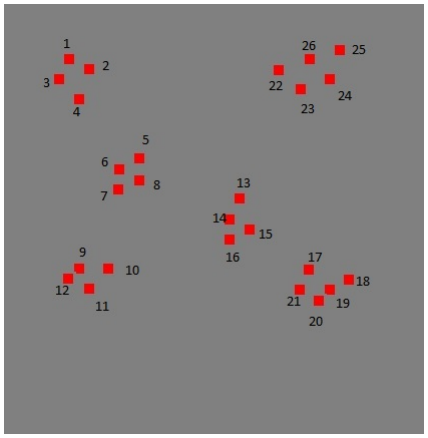

Figure 3. Set 5, the respondents had to identify nr 6. GBLA: ***The square in the left top in the topmost group of four, below the leftmost group of four.*** **SoG:** ***The topmost square below the topmost square below the bottommost square in the leftmost group of four.***

evaluations both had a different order of sets. Only the example question was always in the beginning.

These evaluation were made using the ThesisTools[5] website and distributed at random over the internet. ThesisTools did not support the measuring of time on page, therefore the respondents were asked to keep track of the time themselves. Furthermore the respondents were asked to provide their own description of the square they just identified. The resulting descriptions are further discussed in section 4.

3.2 Results

The survey was filled in more than anticipated. In total 59 people (partially) filled one of the two survey's. Unfortunately only 28 of the respondents completely filled all the forms with useful data. Two of these 28 filled in all but the final two questions. One in each survey. These two missing answers have been counted as if the answer was wrong, the time measured at these two questions is not used for the averages.

The average age of the respondents was 24.5. The male:female ratio was 8:1 and most had (or are doing) a bachelor degree or higher education level. Every one of the 28 respondents came to the right entity in the example question, giving it a 100% score. With this information it can be argued that every respondent understood the questions and knew what was asked of him/her.

The results of the other sets (2-10) can be found in figure 2 and 5.

3.2.1 Accuracy

The accuracy is number of good answers divided by the number of respondents. This always gives a value between zero and one. The total accuracy of the GBLA expressions with 78% is a tad higher than that of the SoG algorithm (66%), this information is sufficient to base conclusions on. The survey results follow binomial distribution. The 99% ($\alpha = 0.01$) confidence interval for the GBLA expressions is [0.685, 0.875] where $\hat{p} = 0.78$ and $n = 125$[6]. Since the accuracy of SoG (0.66) is not in the confidence interval, it is at least 99% sure that the GBLA expressions perform better than the SoG expressions.

Some of the sets show some interesting results that warrant some deeper analysis.

- Set 2, 4, 5 and 6 show a clear victory for the Group-based algorithm. Especially set 5 (see figure 3) where the Group-based algorithm scored a staggering 100% and the SoG only 46%. The poor performance of the SoG algorithm can be explained by the nature of that context set. It had lots of entities in relatively large groups. Therefore the number entities you'd had to go through using the SoG algorithm was quite large, while the GBLA could do with just the groups, making the expressions much more understandable.

- Set 3 and 8 show a slight advantage for the SoG algorithm. In set 3, the SoG description was quite a bit shorter than that of the GBLA. Furthermore, the GBLA description had a "counter intuitive" relation, namely the group "most right" left of the "rightmost" group, switching right and left three times in the expression. The expression generated by GBLA in set 8 was also found to be quite a bit harder than its SoG counterpart. The SoG expression was indeed simpler because it was able to use a more direct relation. It shows that in some situations the SoG has an advantage over GBLA because the group layout favours relations between entities instead of groups. This is especially the case if there is some entity in a group that can be identified in one go, that is relatively close (relation wise) to the target entity.

- Set 9 (see figure 4) was especially hard it seems. The explanation for the poor performance of the GBLA can be found in the group of 4 in right. At least 46% of the people identified that group as two groups of 2, choosing number 13 as a result. This ambiguity in definition of groups can cause bad expressions, but this is not necessarily the fault of the GBLA. If you just count the following to the intended group, then you could argue that the accuracy in total is 73% instead of 27%. Also interesting are the results of the SoG algorithm in this set. The respondents chose number one (although less than the "right" choice) quite often, which is also quite understandable since it is in the same relation to number 6 as 7 is, for some reason they counted it on the same level as 7,14 and 13.

[5] www.thesistools.com

[6] The total number of GBLA expressions evaluated by the 28 respondents.

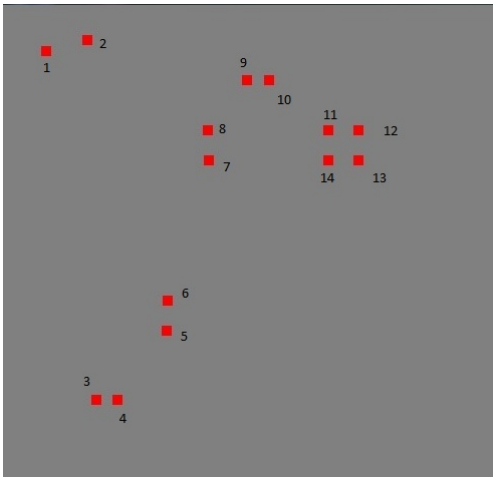

Figure 4. Set 9, the respondents had to identify nr 7. The GBLA expression: ***The bottommost square in the group of two, that is the most right below the topmost group under the topmost group of two.***

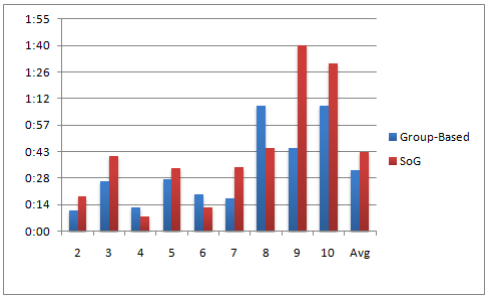

Figure 5. The average time respondents used for the questions. Lower is better.

3.2.2 Time

The time taken to identify seems to have a corrolation with the accuracy. In total the group-based expressions used less time than the SoG expressions, but also here there are some interesting exceptions to the general result.

- Sets 2, 3, 4, 5, 6 and 7 all took very low amount of time to complete. Some even within 10 seconds. The SoG expressions have some that take quite a bit more time than the GBLA expressions, such as 3 and 7. Interesting to see however, is that exactly those two expressions have a higher accuracy for the SoG algorithm. It might be the case here that although the expression itself seems harder to understand, hearers look at it more closely and more often than not come to the right entity. Although the raw data for SoG suggest that the people that took the longest[7] also got to the wrong answer, while the data for GBLA shows that the people that think a bit longer[8] often come to the right conclusion.

- Set 8 is interesting because it shows that GBLA expression took quite a bit longer than the SoG counterpart. This concurs with the observation in the previous section, that the GBLA expression for that set was quite suboptimal.

- Set 9 and 10 show a much higher time for the SoG expressions than for the GBLA expressions. Interestingly, SoG performed better accuracy wise for set 9, but respondents took on average twice as much time. Again, the people that took the most time actually had it wrong, while the wrong answers in GBLA did not take significantly more time than the others.

It seems that for the SoG expressions, the people that did not understand them, did not notice their "error" and continued to puzzle (taking a long time) to finally come to the wrong answer. While in the GBLA expressions it doesn't really matter if you come to the right or wrong entity for the time it takes you to get there.

Since there is such a large correlation between time taken and the coming to the correct answer, variables other than the use of SoG and GBLA come into play. Therefore we cannot make a definitive conclusion. Important to note that the time values used here were not measured but filled in by the respondent, therefore these times should not be taken too precise. If we filtered the respondents on having the right answer or not, we might get only the people that are fond of some type of expression and therefore influencing the results. Also the total time difference between SoG and GBLA does not seem significant enough to say something definitive about it (only 10 seconds difference).

[7]Both wrong answers took about 1:30.
[8]Two of the three that had it wrong were in the fastest 50%.

4. DISCUSSION AND FUTURE WORK

In this section some interesting observations during the evalution are discussed. Also Albert de Graaf [2] did a similar evaluation and his results are discussed as well. Finally there are some things that could be improved upon in future work in the future work section.

4.1 Discussion

The evaluation measured two different aspects, namely the accuracy and the time. But apart from that the respondents were asked to describe the just identified entity themselves. This was purely to see interesting differences and gain ideas for future improvements. It should be noted however that every respondent saw the expressions generated by the SoG algorithm and the GBLA, and is therefore possibly influenced by those strategies. Therefore it could be that some respondents would have given totally different expressions if they did not see the SoG or GBLA expressions first. Furthermore not all expressions uniquely identify the intended entity.

Even so, some interesting patterns emerge from their input.

- A lot of people use expressions like "the second group or entity from the right" instead of the expression generated. Interestingly since some of the expressions could easily be simplified that way, especially the expressions that are "counter intuitive". It might be so that some hearers prefer to hear a coordinate like system over, what I'd consider, a more natural way using group references. This does not stop at two or three, there are respondents that continue using this strategy for things like "the 11th square from the top" . This method is probably quite accurate in most cases, it could possibly take a long time if the number of elements in the context grow.
- The use of middle. Although stated before that determining the middle is not so trivial and explicitly avoided for this evaluation, it is still interesting to see that many people use that kind of expression to make an expression.
- The use of shapes or combining groups to make new or bigger and unique groups. Set 9 for example had three sets of two entities at the same height. Some of the respondents combined these three sets of two and combined referred to them as the cluster of six. Incorporating such strategy into an algorithm does not seem so trivial.
- Not only using a coordinate system for groups, but stating things like the "most right square in the middle row" suggest a system with complete disregard for the locations of all other squares not in that row. This strategy seems somewhat hard to practice if the context set is not so neatly structured as they were in the evaluation. The context sets were in a grid layout, therefore there was no overlap and the respondent could use the rows.
- Also interesting to see is that although respondents talk about the same square, some refer to some group as the middle group while others refer to that same group as the bottom group. Exactly the reason why using the middle was avoided.

Altogether quite some interesting methods to describe the same entity. While most seem somewhat impractical and are probably of no real use in algorithms it is still important to see what humans would produce in these kind of situations. [8]

Albert de Graaf [2] used the same scenes as the evaluation in this paper to try to find out if iterated versions of the generated expressions would be better understandable than the expressions as I presented them. In specific meant for text to speech applications. The respondents in his evaluation could hear the expression as many times as they wanted. Interestingly the expressions (both iterated and non-iterated) scored quite high on accuracy, while he used the same expressions as this paper. Even more interesting are the results for the time measured in the experiments. He proved that there was a significant improvement in time by using the iterated version; also the time of the non-iterated version seems to be better than the time filled in by the respondents in this paper. The speed and accuracy differences between our two experiments seem significantly enough to suggest that the method of presentation helps respondents understanding the expression. It might be worth it to spend some time on the presentation of the expressions, rather than focusing on the expression itself.

4.2 Future work

During the evaluation a less than optimal performance was visible when referring to groups in a particular manner. This "counter intuitive" expression makes the hearer switch his/her directional focus multiple times in one expression, in particular the opposite direction. This causes them to lose track of the expression and failing to identify the entity. These types of expressions are quite easy to detect however, furthermore, they are easy to avoid. Instead of making an expression like "the rightmost group left of the rightmost group" we could just refer to "the second group from the right" in these situations instead. As the discussion above shows it is quite common for humans to produce as well.

Another improvement of the algorithm lies within the used relations. More specific relations could help reduce the difficulty of the expressions. The relations used in the version that produced the expressions for the evaluation, only four kinds of relations were used. Namely "right, left, above and below" . If these are extended with the logic to use directly right, left, above and below to indicate a more narrow band in which the other group can be. Also adding combinations of existing relations, such as "right above, left above" could significantly limit the number of relations needed before arriving at the target group. The Basic Group Incremental Algorithm should also be improved with these kind of locational properties, so it could identify groups with expressions as "the group in the top-right".

Maybe the most important improvement could be made in the group detection algorithms. This algorithm uses some sort of visual perceptive group detection algorithm. If these are improved in such a way that they can detect lines and/or shapes, this information could be used to identify many more groups without relations, as result giving the GBLA more groups to relate the target group to. [6, 7]

5. CONCLUSION

In the introduction of this paper we stated the following research questions.

How can we make an algorithm that can produce referring expressions with groups as landmarks that are understandable by humans? To be able to answer the main research question a number of sub-research questions were answered.

- To what extent are the generated referring expressions understandable?
- Are referring expressions with groups as landmarks preferred over referring expressions with a single object as a landmark? If so, to what extent?
- In what context-sets are referring expressions with groups as landmarks suitable? And how can these situations be identified?

The paper describes the GBLA algorithm using the Locative algorithm [4] as basis for referring expressions with groups as landmarks. So to what extend are the expressions generated by the GBLA algorithm understandable?

The evaluation showed that the expressions used in the evaluation had an accuracy of 78%. Of course this is very dependent on the sets for which these expressions are generated. When looking at the results, especially the expressions of difficulty degree 2 (see Section 3) had very high accuracy ratios. The evaluation also compared the expressions with expressions for the same entities generated by the SoG [5] algorithm. The difference in accuracy was enough to say with 99% certainty that the GBLA expressions perform better than the SoG expressions.

This conclusion also brings us to the next point. Are referring expressions with groups as landmarks preferred over referring expressions with a single object as a landmark? Generally speaking, yes. Most of the sets showed that using groups results in faster identification times and higher accuracy. However not in all context sets the GBLA expressions perform better.

Examining the results from the evaluation shows that the places where it went wrong are most likely because some groups were differently identified than the expressions generator though they would be. This especially happened in Set 9 (see section 3.1) where the group of four was identified as two groups of two. This identification process is probably the result of the fact that all other groups in the context set are clearly groups of two, therefore, people probably reason, that group of four must also consist of two groups of two. Furthermore, the expression was formed in such a way that it did not use the group of four, and therefore did not correct the vision people had of that group, resulting in very poor performance.

Ideally, these kinds of dangerous sets should somehow be identified beforehand, although it is not really clear how this should happen and what the algorithm should do with this information. This problem should be looked at in more depth in future research.

The main research question is answered by section two. When the GBLA algorithm is used, unfortunately not all of the generated expressions are understandable for everyone, but in general they should be more understandable than expressions generated by other algorithms, especially for difficulty two type of expressions.

6. ACKNOWLEDGMENTS

I am grateful to all the respondents for their input and feedback. Thanks to Albert de Graaf for his constructive feedback during the research and the information he provided with his research. Special thanks to Mariët Theune for her guidance and advice throughout the course of this research.

7. REFERENCES

[1] R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[2] A. de Graaf. Iterations in referring expressions. In *Proceedings of the* 15^th *Twente student Conference on IT*, 2011.

[3] A. Gatt and A. Belz. Introducing Shared Tasks to NLG: The TUNA Shared Task Evaluation Challenges. *Empirical Methods in Natural Language Generation*, pages 264–293.

[4] J. D. Kelleher and G.-J. M. Kruijff. Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1041–1048, 2006.

[5] F. Kotaro, W. Satoru, and T. Takenobu. Group-based generation of referring expressions. In *Proceedings of the Fourth International Natural Language Generation Conference*, INLG '06, pages 73–80, 2006.

[6] F. Landragin, N. Bellalem, and L. Romary. Visual salience and perceptual grouping in multimodal interactivity. In *First International Workshop on Information Presentation and Natural Multimodal Dialogue*, pages 151–155, 2001.

[7] K. Thórisson. Simulated perceptual grouping: An application to human-computer interaction. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 876–881, 1994.

[8] J. Viethen and R. Dale. Algorithms for generating referring expressions: do they do what people do? In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 63–70. Association for Computational Linguistics, 2006.