Generating Quantified Descriptions of Abstract Visual Scenes

Guanyi Chen[†], **Kees van Deemter**[†‡], **Chenghua Lin**[*]
[†]Department of Information and Computing Sciences, Utrecht University
[‡]Department of Computing Science, University of Aberdeen
[*]Department of Computer Science, University of Sheffield
{g.chen, c.j.vandeemter}@uu.nl, c.lin@sheffield.ac.uk

Abstract

Quantified expressions have always taken up a central position in formal theories of meaning and language use. Yet quantified expressions have so far attracted far less attention from the Natural Language Generation community than, for example, referring expressions. In an attempt to start redressing the balance, we investigate a recently developed corpus in which quantified expressions play a crucial role; the corpus is the result of a carefully controlled elicitation experiment, in which human participants were asked to describe visually presented scenes. Informed by an analysis of this corpus, we propose algorithms that produce computer-generated descriptions of a wider class of visual scenes, and we evaluate the descriptions generated by these algorithms in terms of their correctness, completeness, and human-likeness. We discuss what this exercise can teach us about the nature of quantification and about the challenges posed by the generation of quantified expressions.

1 Introduction

A long tradition of research in formal semantics studies the question how speakers express quantification. Much of this work starts from the idea that the prime function of Noun Phrases (NPs) is to express quantitative relations between sets of individuals. Obvious examples are akin to the universal and existential quantifier of First Order Predicate logic, as in "*all A are B*" and "*Some A B*". Crucially, researchers observed that NPs permit much more variation in terms of the relations expressed: not only can we say things like "*between 2 and 5 A are B*" (where "*between 2 and 5 A*" is a noun phrase), but also things like "*most A are B*" and "*Few A are B*", which are not even expressible in First-Order Predicate Logic. For this reason, this research are, is known as the study of Generalised Quantifiers, often regarded as starting from Mostowski (1957); the link with natural language was more fully established with Barwise and Cooper (1981) and further elaborated in Keenan and Moss (1985); Van Benthem et al. (1986); Peters et al. (2006).

Though the main focus of this work has been on mathematical logic, there is a growing body of empirical work. For example, there is some psycholinguistic work on the human use of vague quantifiers (Moxey and Sanford, 1993), and work that investigates the links between quantifiers' logical types and the human processing of quantified expressions (Szymanik and Zajenkowski, 2010; Szymanik et al., 2016, QEs). Sorodoc et al. (2016) looked at speakers' choice between *all*, *some*, and *no* (see also Grefenstette (2013) and Herbelot and Vecchi (2015)). However, there has been no attempt to find out how the wider class of all ("generalised") quantifiers are used by human speakers.

In a simple version of the problem, consider a table with two black tea cups and four coffee cups, three of which are red while the remaining one is white. Each of (a)-(d) describes the scene truthfully (though not necessarily optimally):

(a) *There are some red cups.*
(b) *At least three cups are red.*
(c) *Fewer than four cups are red.*
(d) *All the red objects are coffee cups.*

The computational challenge is to design an NLG algorithm that chooses the "best" statement, choosing from all the ones above and many others. In more complicated situations, which cannot be adequately described by one single quantified pattern, the algorithm should generate a sequence of sentences. This is a difficult *computational modelling* challenge to which the present paper makes a contribution.

In section 2, we discuss a strand of work in

NLG that provided inspiration for this enterprise. Section 3 proposes a computational mechanism, which will show up some surprising commonalities between the task of the present paper – Quantified Description Generation (QDG) – and the Referring Expressions Generation (REG). Section 4 details our evaluation experiment and section 5 discusses implications and future work. Code for the proposed QDG algorithms can be found at: https://github.com/a-quei/quantified-description-generation.

2 Background

An early investigation of the choice between patterns (a)-(d) above rested on the idea that the generator should always choose the logically strongest statement that holds true in the situation described (Creaney, 1996). Thus, statement (b) was always preferred over (a). However, this attractive idea ran into difficulties over pairs of statements that are logically independent of each other, such as (b) and (c), or also (b) and (d), where each of the two statements conveys some information that the other one does not. Clearly, other computational mechanisms are called for. The present paper proposes and evaluates a family of such mechanisms.

A body of work that is indirectly relevant concerns the generation of *referring* NPs (Krahmer and van Deemter, 2012; van Deemter, 2016). One strand of this work focusses on corpora of referring expressions (REs) that were elicited under experimentally controlled conditions (e.g., the TUNA corpus (Gatt et al., 2007; van Deemter et al., 2012); such corpora were used in evaluation campaigns where computer-generated referring expressions were compared with the corpus gold standard (Gatt and Belz, 2010). This comparison allowed researchers to know which algorithms worked best, and to develop new algorithms that match human language production even more closely. These evaluation campaigns created a tradition of NLG "generation challenges", consistent with the broader tradition of evaluation campaigns in Machine Translation (Barrault et al., 2019), parsing (Buchholz and Marsi, 2006), and other areas of Computational Linguistics.

Focusing on a far wider class of NPs, a series of elicitation experiments was recently conducted, called here the QTUNA experiments (Q stands for quantification). We will summarise the main findings from these experiments before proposing and experimentally comparing algorithms that seek to mimic the aforementioned corpus. More details can be found in Chen et al. (2019) [1].

The QTUNA experiment was set up in order to study how speakers use sequences of QEs to describe a visual scene. Participants were asked to describe a series of abstract visual scenes. Participants were told that their descriptions should allow readers to *reconstruct* the scene *modulo* their location. Each scene contained n objects, which is either a circle or a square and either blue or red. To chart how domain size n influences human production of QEs, QTUNA conducted 3 experiments, with n of 4, 9, and 20 respectively.

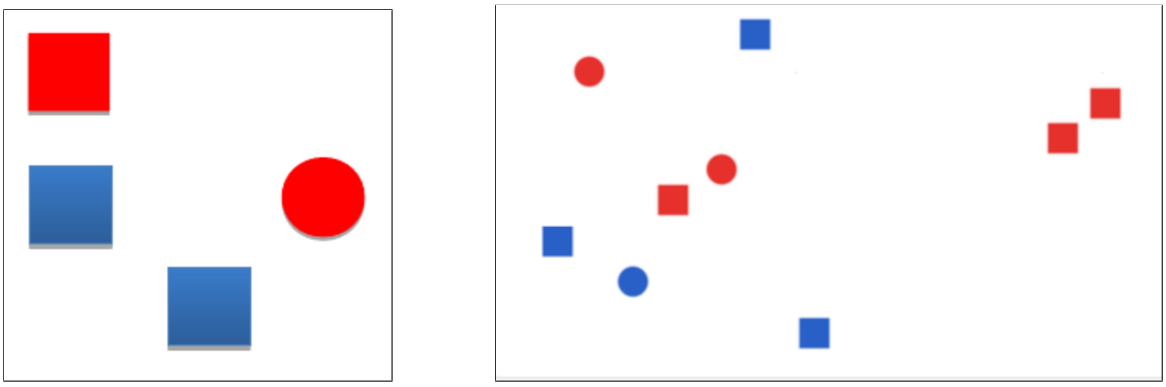
The experiment was conducted on a total of 187 subjects, producing a total of 1414 multi-sentence descriptions for the resulting QTUNA corpus. Each description was annotated with its meaning representation: following Barwise and Cooper (1981), each quantifier was cast as a relation between 2 or more set-denoting arguments. For example, "*Half of the objects are blue*" was represented as Half(O, B). [2] To represent plurality, (as in "*Some blue squares ...*", as opposed to "*Some blue square ...*") the suffix *-s* was appended. Examples with their corresponding meaning representations are shown in Table 1.

Analysis of this corpus (see Chen et al. (2019)) taught us the following lessons: 1) Speakers use more vague quantifiers (e.g., *most*, *few*) as domain size increases; 2) Speakers also use more under-specifications in larger domains, describing large domain "with a broad brush"; 3) The average length of quantified descriptions is not significantly larger in larger domains than in smaller ones; 4) Speakers tends to start describing the high level information of the whole scene, before going into detail about parts; 5) Speakers tend to mention shape before the colour.

Quantified Expression Generation. Previous computational models of speakers' choice of quantifiers focus on the choice between a limited number of candidate quantifiers (Grefenstette, 2013; Yildirim et al., 2013; Herbelot and Vecchi, 2015; Sorodoc et al., 2016; Castillo-Ortega et al., 2009; Barr et al., 2013; Ramos-Soto et al., 2016). Barr et al. (2013) studies the use of quantifiers that

[1] The dataset can be found at: https://github.com/a-quei/qtuna

[2] O, S, C, R, C are property representations representing object, square, circle, red, and circle, respectively. BS means blue square, and so on.



(a) (b)

Figure 1: Examples from (a) the $n = 4$ experiment; (b) the $n = 9$ experiment.

n	Description	Meaning
4	*There are 4 squares. All objects are blue.*	$\exists_{=4}(S_s) \wedge \text{All}(O_s, B_s)$
9	*Most of the items are red circles, but there are a couple of blue squares.*	$\text{Most}(O_s, RC_s) \wedge \exists_{=2}(BS_s)$
20	*All the objects in the picture are circles and majority of them is blue.*	$\text{All}(O_s, C_s) \wedge \text{Majority}(O_s, B_s)$

Table 1: List of example descriptions from QTUNA corpus; n indicates domain size.

are part of a referring expression. The present papers models how human speakers use quantifiers as part of a wider task, namely the task of describing a complicated visual scene.

3 Quantified Description Generation

The aim of our generator is to perform the same task as was given to the human participants in the QTUNA experiments, but unlike those experiments, we would like our algorithm to succeed in domains of any reasonable size – including the sizes 4, 9, and 20 of those experiments and all sizes in between. Domains with fewer than 4 objects, and those in which there are many more objects than can be counted in a few seconds, are beyond the scope of this work. Participants in the QTUNA experiments were asked to make the generated descriptions correct (i.e., truthful) and complete (i.e., giving as much information as can reasonably be expected). Our generator endeavours to do the same and will therefore be evaluated using these same two criteria, plus an additional criterion that asks more explicitly how human-like the generated descriptions are. We start by introducing the general framework of our QDG system, which is a pipeline including a pre-processor, a QDG algorithm, and a surface realiser. Second, we introduce two pre-defined knowledge bases that will be used by the QDG system. Third, we propose and sketch two QDG algorithms.

3.1 General Framework

Our explanation will make use of the following formalisation. Given a target scene s with its domain knowledge $\mathcal{K}_d$ (for details see §3.2), the generator constructs a set S containing all possible scenes in the same domain as s. The generator then calls a QDG algorithm to construct a description $\mathcal{D}$ containing a set of L QEs $\{q_l(v)\}_{l=1}^{L}$. We use $q(\cdot)$ to represent a quantified pattern, for instance $\text{All}(\cdot,\cdot)$. Furthermore, v is a property tuple. If v is capable of filling in the slots of a quantified pattern, we say that the pattern *accepts* v, and we write $q(v)$. The QE $q(v)$ is a logical form, as introduced in §2. The algorithm selects from a set of candidate patterns $\mathcal{Q}$, based on the common knowledge $\mathcal{K}_c$ (for details see §3.2) defined on $\mathcal{Q}$ by mimicking how human beings did so in the experiment. Finally, a simple template-based surface realiser is called to map $\mathcal{D}$ into natural language text. The architecture of the generator is shown in Figure 2.[3]

[3] In the future, a Sequentialiser will put the QEs in an optimal order, but the algorithms in this section will realise the

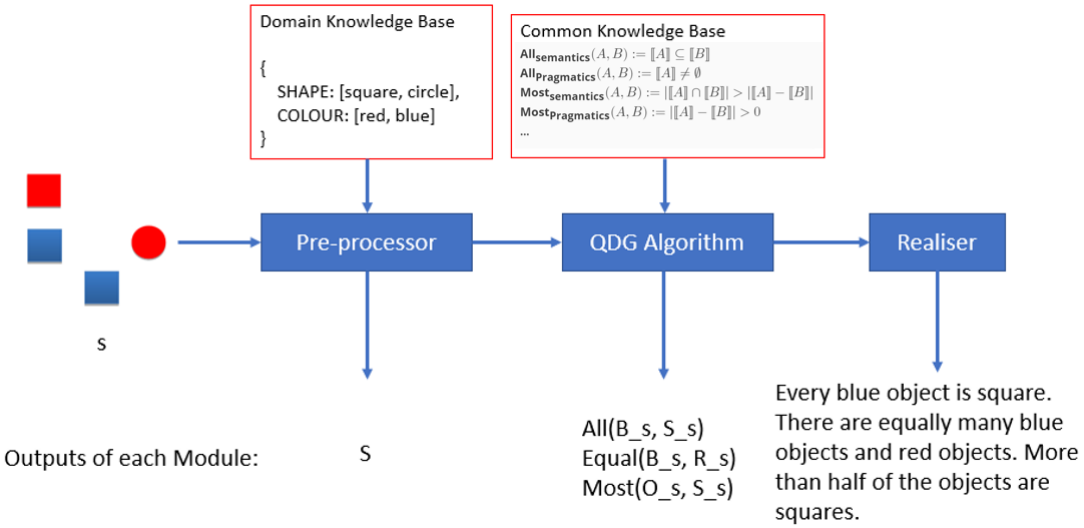


Figure 2: Sketch of the Quantified Description Generation System.

3.2 Knowledge Bases for QDG Algorithms

Domain Knowledge $\mathcal{K}_d$**.** This is the list of possible attributes with their possible values, represented as a set of key-value pairs. For example, matching the experimental setting of QTUNA, its domain knowledge is $\{\texttt{SHAPE} : [square, circle], \texttt{COLOUR} : [red, blue]\}$.

Common Knowledge $\mathcal{K}_c$**.** This is a body of knowledge that corresponds to the quantified patterns in Q. For a quantified pattern $q(\cdot)$, this knowledge includes the meaning of the quantified pattern, and a list of possible property tuples that could be assigned to v. The list of supported patterns can be found in the supplementary materials.

The meaning of a quantified pattern has two parts: its semantics and its pragmatics. For example, the *semantics* of $\text{All}(A, B)$ asserts that $[[A]] \subseteq [[B]]$. The *pragmatics* says that $[[A]]$ is not empty. Determining the semantics and pragmatics of each English quantifier term can be tricky, but the QTUNA corpus allowed us to choose definitions that match majority usage in that corpus.

3.3 An Incremental QDG Algorithm

Looking at the QTUNA dataset, some quantifier patterns are more frequent than others, and some choices of properties to fill a given pattern are more frequent than others. Therefore, to mimic people's production of quantified descriptions, a natural idea is to maintain a sequence of properties and a sequence of fillers, which the algorithm can then make use of to determine in what order to consider the different types of statements for inclusion in the generated description:

- **Quantifier Sequence.** Inspired by our finding that humans tend to start describing the scene as a whole, QEs such as *all*, *half*, and *most* have high priority.
- **Property Sequence.** Informal inspection of QTUNA suggested that, for patterns of the form $\text{All}(A, B)$, the first argument, A, is more often a SHAPE property, whereas B is more often a COLOUR. For example, the algorithm should check the property tuple (S, R) before (R, S).

The algorithm generates the description by considering possible quantifiers and properties one by one, starting at the top of the sequence, working its way down. Because of its similarity with the Incremental algorithm of Dale and Reiter (1995), we call this algorithm the *Incremental QDG algorithm* (abbreviated QDG-IA). Likewise, we allude to the "Preference Order" of properties employed by Dale and Reiter's REG algorithm by speaking of the Quantifier Preference Order (instead of Quantifier Sequence) and the Property Preference Order (instead of Property Sequence). Algorithm 1 shows the detailed QDG-IA.

Given the inputs listed in Algorithm 1, the QDG-IA will go through all the quantified patterns in $\mathcal{Q}$ in the order of the quantifier preference order. In each iteration, for the selected quantified pattern

QEs in the order in which they were selected.

Algorithm 1 The Incremental Algorithm for Generating Quantified Descriptions

Input: A target scene s, a set $\mathcal{S}$ of all possible scenes, a set of quantified patterns $\mathcal{Q}$, the common knowledge $\mathcal{K}_c$ defined on $\mathcal{Q}$, a Quantifier Preference Order defined on $\mathcal{Q}$, a set of all possible property tuples in the domain $\mathcal{V}$, and a property preference order defined on $\mathcal{V}$.

Output: A quantified description $\mathcal{D}$ of s that uses conjunctions of single or multiple $q(v)$s.

1: $\mathcal{D} \coloneqq \{\}$
2: **for** each q in $\mathcal{Q}$ (in order of the Quantifier Preference Order) **do**
3: **for** each v in $\mathcal{V}$ such that q accepts v (in order of the Property Preference Order) **do**
4: $q(v) \coloneqq \texttt{pluralise}(q(v), s)$
5: **if** $q(v)$ is true for s, and $\mathcal{D} \not\models q(v)$ **then**
6: $\mathcal{D} \coloneqq \mathcal{D} \cup \{q(v)\}$
7: $\mathcal{S} \coloneqq \{s' \in \mathcal{S} : q(v) \text{ is true for } s'\}$
8: Until $\mathcal{S} = \{s\}$ or $|\mathcal{D}| \geq \delta$

$q(\cdot)$, QDG-IA will test all possible property tuples accepted by $q(\cdot)$, in the order of property preference order. (Recall that the information which $q(\cdot)$ accepts which property tuples can be found in the common knowledge $\mathcal{K}_c$.) The algorithm then calls the `pluralise` function on the QE, where the `pluralise` assigns a plural suffix on properties that appear multiple times in the target scene s.

In step 5, the algorithm will first validate whether $q(v)$ is correct as a QE for s. The validation is performed by using both the semantics and pragmatics of $q(v)$, in a somewhat unexpected manner. The semantics of a QE $q(v)$ is the same as the semantics of the pattern $q(\cdot)$, but the pragmatics takes issues such as plurality into account.

For example, consider $Few(O, S)$. The semantics says that fewer than n O are S; the pragmatics of this QE says that there exist *at least two* O. Consequently, the QE will not be included in the description unless there exist two O (and if it is included, this information is taken into account when scenes are removed from $\mathcal{S}$ in line 7).

Subsequently, step 5 ensures that $q(v)$ *does not follow* from the generated description $\mathcal{D}$, ensuring that $q(v)$ rules out one or more further scenes. Crucially, this validation is performed by using only the semantics of $q(v)$, not the pragmatics. To see why, consider QE $Few(O, S)$ again: if the pragmatics of this QE (which says that at least 2 O are S) were taken into account during validation, then the algorithm would end up adding this QE to a description $\mathcal{D}$ *even* in cases where the QE's only contribution to $\mathcal{D}$ is the (pragmatic) requirement that at least 2 O are S (because other QEs, previously added to $\mathcal{D}$, already ensure that fewer than n O are S). This would tend to lead to unwieldy descriptions, some of whose constituent QEs contribute very little to the description of the scene.

Once the above two conditions have been validated, $q(v)$ is appended at the end of the description and the scenes for which $q(v)$ is not true are removed from S. Both semantics and pragmatics are used for removing such "distractor" scenes. The generation terminates when all the distractors are removed from S or the length of D reaches an upper bound δ. The idea of setting a upper bound comes from the observation that, in QTUNA, descriptions were remarkably constant across domain sizes. As for the design of preference orders, we started with testing the following settings, once again based on analysis of the corpus: 1) the quantifier preference order is a linear preference order, which starts with All $\succ$ Half $\succ$ Most [4] (A $\succ$ B means A following B in the preference order); 2) the property preference order was designed by following some constraints, e.g., `SHAPE` properties have higher priorities in the first argument place of a quantified pattern, whereas `COLOUR` properties have higher priority in order argument places. The preference order was designed and refined based on the data from the pilot experiments and the n=4 QTUNA experiment.

When we ran the algorithm, we found that some quantified patterns that have low preference will never be chosen by the algorithm, so the generated descriptions used only a very limited set of patterns. For example, the pattern All$(\cdot, \cdot)$ has higher preference than the pattern Everything$(\cdot)$, and consequently the latter is *never* chosen, because its meaning is covered by the former. To increase the variety of quantifier patterns in the descriptions generated, we introduced a probability θ, with which the QDG-IA can perform a one-off move of a quantified pattern with low preference into a higher preference order. Since quantified patterns with low preference order should not appear frequently, the value θ should not be high. For

[4] A full linear quantifier preference order can be found in the supplementary materials.

Algorithm 2 The Greedy Algorithm for Generating Quantified Descriptions

Input: A target scene s, a set $\mathcal{S}$ of all possible scene, a set of quantified patterns $\mathcal{Q}$, the common knowledge $\mathcal{K}_c$ defined on $\mathcal{Q}$, a set of all possible property tuples $\mathcal{V}$

Output: A quantified description $\mathcal{D}$ of s that uses conjunctions of single or multiple $q(v)$s.

1: $\mathcal{D} \coloneqq \{\}$
2: **while** $\mathcal{S} \neq \{s\}$, and $|\mathcal{D}| < \delta$ **do**
3: $\quad q(v) \coloneqq \texttt{FindBestQE}(s, S, \mathcal{Q}, \mathcal{V}, \mathcal{K}_c)$
4: $\quad$ **if** $\mathcal{D} \not\models q(v)$ **then**
5: $\quad\quad \mathcal{D} \coloneqq \mathcal{D} \cup \{q(v)\}$
6: $\quad\quad \mathcal{S} \coloneqq \{s' \in \mathcal{S} : q(v)$ is true for $s'\}$

the work reported in this paper, we set θ to 0.1. A list of quantified patterns that have high meaning overlap with each other (so the use of one tends to preclude the use of the other) can be found in the supplementary material.

3.4 A Greedy QDG Algorithm

The QDG task can be viewed as a search problem, searching for the best set of QEs that can single out a scene s from its distractors. From this perspective, another natural idea is to use a greedy algorithm, which selects a QE that singles out the largest number of distractors in each iteration.

We propose a greedy algorithm for QDG (abbreviated QDG-GREEDY), as in Algorithm 2. In each iteration, the algorithm calls the function `FindBestQE` to choose the QE that rules out the most distractors from all possible QEs. Specifically, for each QE $q(v)$, it first pluralises the $q(v)$, as in step 4 of Algorithm 1, and check whether it is true for the target scene using $q(v)$'s semantics and pragmatics. Then it calculates the number of distractors that can be removed by only using $q(v)$'s semantics. The number of distractors that $q(v)$ can remove is called $q(v)$'s discriminatory power. At last, the `FindBestQE` will return the QE that has the highest discriminatory power.

In line 4, the algorithm ensures that the discriminatory power of the selected $q(v)$ is not zero. If it passes this test, $q(v)$ is added to $\mathcal{D}$, and distractor scenes are removed from S. The stop criterion of QDG-GREEDY is the same as the one of QDG-IA.

In order to increase the variation in the generated quantified descriptions, the `FindBestQE` function returns all the best QE that has the same discriminatory power. In step 5, one of these is randomly selected and appended to $\mathcal{D}$.

3.5 Realisation

Since our main focus was on quantifier choice (not wording), the QDG system uses a simple template based surface realiser. For each quantified pattern, there is a specific template. When filling the slots with chosen properties, some simple syntactic and morphological operations are employed. For example, if a `COLOUR` property takes the first place of a quantified pattern, a noun is appended to package it into a NP (e.g., *red* $\rightarrow$ *red object*). If a property has a plural suffix, the surface form of the property is mapped into its plural form. A number of further constraints, specific to particular quantifier patterns, were also coded in the realiser.

3.6 Comparing QDG with REG

The two algorithms presented above (1) start out from the set S of all possible scenes, (2) accumulate, one by one, quantified statements $q(v)$ that are true for the target scene and false for some other elements of S, (3) remove from S all scenes for which $q(v)$ is false (4) until the target scene is the only remaining element of S or some other stopping condition is met.

Quantification is not normally seen as a reference problem. Yet our approach resembles the classic algorithms for Referring Expressions Generation (REG) originally discussed in Dale and Reiter (1995), where a referring expression is constructed by accumulating properties (e.g., colours, sizes) one by one, each of which is thought to "remove" from consideration a set of "distractor objects" (i.e., potential referents that differ from the target referent in one or more respects). We have emphasised this similarity throughout, by using terms familiar from REG (e.g., "target", "(removing) distractors", "preference order", and so on).

One difference is that, in REG: (1) a set of potential *referents* takes the place of the set of all possible *scenes*, and (2) properties (such as "red") take the place of quantified statements. These differences have important consequences. Quantified statements are much more complicated than properties, hence the distinction between choosing a pattern $q(\cdot)$ (line 2 of Algorithm 1) and instantiating the pattern by choosing a value v (line 3). And, unlike the set of all possible referents in REG, the set S of all possible scenes is not a given but needs to be computed from the properties that are given

(e.g., red, blue, circle, square). Furthermore, our algorithms have had to find a way to take both the semantics and the pragmatics of a quantifier pattern into account. Finally, the QTUNA experiments have taught us that, except in very small domains (such as the QTUNA one where n=4), vague quantifiers (such as *many*, *most*) are highly frequent, and hence need to be taken into account by any QDG algorithm. By contrast, REG has been able to disregard vague properties for a long time, because in domains consisting of simple objects (such as the TUNA furniture domains, and those of Dale and Viethen (2009), where all properties were crisp and well defined,) they did not play an important role. A related difference is that, for larger domains, it is typically not feasible to produce a description that identifies the target scene completely. This different in REG, where except in very complicated situations (e.g., van Deemter (2016, Chapters 11-15)), producing a distinguishing description (i.e., one that singles out its referent) is par for the course.

4 Evaluation by Human Judgements

Although our algorithms were informed by extensive elicitation experiments, we wanted to gain additional insights into the quality of the generated descriptions. We were curious how humanlike these descriptions were perceived to be; at a more detailed level, we wanted to know how correct and informative these descriptions were thought to be.

Note that these are not things that could easily be measured automatically. Consider the example of the QE $Few(O, S)$ once again. Suppose 5 out of 20 O are S, then is it correct to say that $Few(O, S)$, or does this underestimate the number of O are S? And if $Few(O, S)$ is all that is said about the proportion of O that are S, is this sufficiently informative or not quite? We are not aware of any existing metric or algorithm that would give us a reliable answer.

To get an insight into these difficult issues, we recruited 4 academics from Utrecht University, none of whom had been involved in our research. Two were young lecturing staff in computational linguistics and two were senior lecturing staff in computational logic and formal argumentation. We divided the experiment into experiment A and experiment B. For A, we randomly selected 3 or 4 scenes (10 in total) from each of the 3 sub-corpora of QTUNA, each of which was paired with 3 descriptions: one by QDG-IA, one by QDG-GREEDY, and one selected at random from our corpus. A number of example scenes paired with their descriptions produced by human beings, QDG-IA, and QDG-GREEDY are listed in Table 3.

As for experiment B, to test the generality of our algorithms, we focused on two new domain sizes, namely n=6 and n=16. For each of these we sampled 6 scenes, each of which was paired with 2 descriptions: one by QDG-IA[5], and one by QDG-GREEDY. (Human-generated descriptions were not available in these new domains.) All these 66 scene-description pairs were put together and randomly allocated to our four "judges". Each judged 33 scene-description pairs. Thus, each scene-description pair was judged by two judges. Judges were asked three questions in each case:

- Q1 (Naturalness): On a scale of 1-5, how likely do you think it might be that this description was uttered by a human? [1=very unlikely, 5=very likely]

- Q2 (Informativity): On a scale of 1-5, do you believe the description is as informative as it can be expected to be? [1= description isnt even nearly informative enough, 5= description gives as much information as is possible]

- Q3 (Correctness): On a scale of 1-5, how correct do you consider this description to be? [1= the description is not at all correct, 5=everything the description says is correct.]

The words Naturalness, Informativity, and Correctness were not seen by the judges. Our instructions to them added, "Please note that were mainly interested in the 'logic' of how people describe the scene, and less in the details of the wording, so please disregard minor syntax errors and typos."

Because in Experiment A, question Q1 was asked about a human-produced description as well as two computer-generated descriptions, this setup allowed us to perform what is essentially a **Turing test**.The other two questions offered invaluable formative evaluation.

We formulated a number of hypotheses: 1) humans perform better at naturalness than QDG-IA and QDG-GREEDY; 2) both algorithms perform better at informativity and correctness than humans, because both of them were explicitly

[5]The preference order used in all 3 experiments of this paper was the same.

	Model	Naturalness	Informativity	Correctness
Experiment A	Human	3.45	4.05	4.6
	QDG-IA	2.85	3.95	4.55
	QDG-GREEDY	3.45	3.8	4.8
Experiment B	QDG-IA	3.7	3.8	4.83
	QDG-GREEDY	3.46	4.2	4.83

Table 2: Average scores for each algorithm and for human-produced descriptions, by naturalness, informativity, and correctness as annotated by our four human judges.

designed to optimise informativity and correctness; 3) QDG-IA performs better at naturalness than QDG-GREEDY. We reasoned that, in REG, the incremental algorithm offered greater human-likeness than the greedy algorithm, so why should things be different this time?

Table 2 shows the scores from the judges. Both algorithms performed well, scoring well over 3 in all except one cell, confirming our impression that the descriptions were of respectable quality.

Remarkably, the first hypothesis was rejected; in terms of naturalness, QDG-GREEDY gained the exact same score as the human speakers. QDG-IA had a slightly lower score, but there was no significant difference (tested by a paired t-test). Though "no difference" results always need to be approached with caution, this might be seen as a case of an NLG algorithm passing a Turing test (focussing on a limited type of language use of course, rather than real conversation).

In an effort to understand the low naturalness performance of QDG-IA, we investigated the cases where QDG-IA had particularly low scores. We found that these were almost always descriptions that contain vague quantifiers (e.g., *few*, *most*), where we know that our semantic and pragmatic definitions were especially tentative. What confirmed this suspicion was the observation that vague quantifiers are used disproportionally often in the scenes of Experiment A (where some scenes for $n = 9$ and $n = 20$ were chosen to find out how vague quantifiers are used), and far less in the scenes of Experiment B (which were generated at random); accordingly, the QDG-IA scored much better on naturalness in Experiment B.

Our analysis of the second hypothesis shows some of the hidden difficulties of the description task that our algorithms solve. All quantified descriptions performed similarly in terms of informativity and correctness. To understand why, we decided to separately calculate the average informativity score for those descriptions in experiment B that were *logically complete* (i.e., the algorithm stopped when $S = \{s\}$). For this reduced set of descriptions, the average scores for QDG-IA and QDG-GREEDY were a mere 3.88 and 4.1, instead of 5 (as one might expect). One possible cause is the fact that our algorithms judged the logical completeness of these descriptions by taking both their semantics and their pragmatics into account, which is something our judges may have disagreed with. Similar things may have happened when they were judging correctness.

The last hypothesis was rejected, as there was no significant difference between the naturalness performance of QDG-IA and QDG-GREEDY. This may be caused by the fact that the preference order that we proposed for quantified patterns has much higher complexity than that of properties (or attributes) in REG. In particular, the number quantifiers is considerable, and our preference order of quantifiers was not linear (as discussed in §3.3). And although our preference orders were informed by a study of the QTUNA corpus, this step was far from water tight, and further improvements to these preference orders are likely, for example if they can be learnt automatically.

5 Discussion and Future Work

We have investigated Generation of Quantified Descriptions of abstract scenes that are populated by a limited number (4-20) of simple objects, and given that this task is full of hidden complexities, our algorithms performed remarkably well. We're curious whether the essence of the findings reported in this paper will stand up when more naturalistic scenes are described. Naturalistic scenes permit many more than 2 attributes (in our experiment: COLOUR and SHAPE), each of which will tend to have more than 2 values. Furthermore, in

Scene	Model	Description
BS:2 RS:2 BC:0 RC:0	Human	All the objects are squares and half of them is blue.
	QDG-IA	Every object is square. There are equally many blue squares and red squares.
	QDG-GREEDY	Half of the objects are blue squares, the rest are red squares.
BS:2 RS:2 BC:5 RC:0	Human	Two objects are red squares. Two objects are blue squares and the remainder is blue.
	QDG-IA	Every circle is blue. Half of the squares are blue. More than half of the objects are blue circles.
	QDG-GREEDY	Half of the squares are red, the rest are blue. Most of the objects are blue circles.
BS:1 RS:6 BC:2 RC:0	Human	Two thirds of the objects are red squares. The remaining objects are blue, of which two are circles.
	QDG-IA	All of the circles are blue. There are more red squares than blue circles. There are more red objects than blue objects. More than half of the blue objects are circles.
	QDG-GREEDY	More than half of the objects are red squares. There is only one blue square. Some objects are blue circles. All of the circles are blue.
BS:9 RS:2 BC:8 RC:1	Human	There is a mixture of squares and circles. Most of them are blue. Some of them are red.
	QDG-IA	All possible objects are shown. A minority of the objects are red squares. Less then half of the objects are blue circles. Less then half of the objects are blue squares. Less then half of the objects are circles.
	QDG-GREEDY	All possible objects are shown. A minority of the objects are red squares. Less then half of the objects are blue circles. Less then half of the objects are blue squares. Less then half of the objects are circles.

Table 3: Examples of quantified descriptions produced by humans, QDG-IA, and QDG-GREEDY. The numbers after each type of object represents the number of that object in the input scene.

real life, it may often be difficult to state what the set S of all possible scenes is, which played such a key role in our algorithms.

Suppose, for instance, you want to describe the people in a football stadium, saying "*About half the people were wearing a hat*". It is then unclear what were all the possibilities that your description is trying to rule out, since it is difficult to determine all the things people *might* be wearing. A possible solution is to abandon the idea of starting from the set of *all* possible scenes, starting instead from a suitably sized *sample* of possible scenes, possibly gleaned from other football matches in the same stadium, proceeding as before in other ways (e.g., terminating when all distractor scenes from the sample have been ruled out). An added advantage of this approach would be that it would be sensitive to constraints and statistical regularities that the speaker and hearer are attuned to. For instance, the sample would tend to bear out the regularity that if one's left shoe is brown then so is one's right shoe.

We have seen that our algorithms for the generation of quantified descriptions bear some striking resemblances with existing algorithms designed for a very different problem, namely REG. We plan to look at some recent advances of REG algorithms, such as those of Ramos-Soto et al. (2016); Monroe and Potts (2015); Li et al. (2018); van Gompel et al. (2019) to see how they can inspire improvements of our QDG models.

Acknowledgements

We thank the reviewers for their helpful comments. We thank Larry Moss, Jakub Szymanik, and Camilo Thorne for suggestions that helped shape our work. Guanyi Chen is supported by China Scholarship Council (No.201907720022).

References

Dale Barr, Kees van Deemter, and Raquel Fernández. 2013. Generation of quantified referring expressions: Evidence from experimental data. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 157–161, Sofia, Bulgaria. Association for Computational Linguistics.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. In *Philosophy, language, and artificial intelligence*, pages 241–301. Springer.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Rita Castillo-Ortega, Nicolás Marín, and Daniel Sánchez. 2009. Fuzzy quantification-based linguistic summaries in data cubes with hierarchical fuzzy partition of time dimension. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 578–585. Springer.

Guanyi Chen, Kees van Deemter, Silvia Pagliaro, Louk Smalbil, and Chenghua Lin. 2019. QTUNA: a corpus for understanding how speakers use quantification. In *Proceedings of the 12th International Conference on Natural Language Generation*.

Norman Creaney. 1996. An algorithm for generating quantifiers. In *Eighth International Natural Language Generation Workshop*.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263.

Robert Dale and Jette Viethen. 2009. Referring expression generation through attribute-based heuristics. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 58–65, Athens, Greece. Association for Computational Linguistics.

Kees van Deemter. 2016. *Computational models of referring: a study in cognitive science*. MIT Press.

Kees van Deemter, Albert Gatt, Ielka van der Sluis, and Richard Power. 2012. Generation of referring expressions: Assessing the incremental algorithm. *Cognitive science*, 36(5):799–836.

Albert Gatt and Anja Belz. 2010. Introducing shared tasks to nlg: The tuna shared task evaluation challenges. In *Empirical methods in natural language generation*, pages 264–293. Springer.

Albert Gatt, Ielka van der Sluis, and Kees van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 49–56, Saarbrücken, Germany. DFKI GmbH.

Roger van Gompel, Kees van Deemter, Albert Gatt, Rick Snoeren, and Emiel Krahmer. 2019. Conceptualization in reference production: Probabilistic modeling and experimental testing. *Psychological review*, 126(3):345.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 1–10, Atlanta, Georgia, USA. Association for Computational Linguistics.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 22–32, Lisbon, Portugal. Association for Computational Linguistics.

Edward L Keenan and Lawrence S Moss. 1985. Generalized quantifiers and the expressive power of natural language. In *Generalized quantifiers in natural language*, volume 4, pages 73–124. Foris Dordrecht.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Xiao Li, Kees van Deemter, and Chenghua Lin. 2018. Statistical NLG for generating the content and form of referring expressions. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 482–491, Tilburg University, The Netherlands. Association for Computational Linguistics.

Will Monroe and Christopher Potts. 2015. Learning in the rational speech acts model. *CoRR*, abs/1510.06807.

Andrzej Mostowski. 1957. On a generalization of quantifiers. *Fundamenta Mathematicae*, 44(2):12–36.

Linda M Moxey and Anthony J Sanford. 1993. *Communicating quantities: A psychological perspective.* Lawrence Erlbaum Associates, Inc.

Stanley Peters, Dag Westerstahl, and Dag Westerståhl. 2006. *Quantifiers in language and logic*. Oxford University Press.

Alejandro Ramos-Soto, Alberto Bugarín, and Senén Barro. 2016. Fuzzy sets across the natural language generation pipeline. *Progress in artificial intelligence*, 5(4):261–276.

Ionut Sorodoc, Angeliki Lazaridou, Gemma Boleda, Aurélie Herbelot, Sandro Pezzelle, and Raffaella Bernardi. 2016. "look, some green circles!": Learning to quantify from images. In *Proceedings of the 5th Workshop on Vision and Language*, pages 75–79, Berlin, Germany. Association for Computational Linguistics.

Jakub Szymanik and Marcin Zajenkowski. 2010. Comprehension of simple quantifiers: Empirical evaluation of a computational model. *Cognitive Science*, 34(3):521–532.

Jakub Szymanik et al. 2016. *Quantifiers and cognition: Logical and computational perspectives*, volume 96. Springer.

JFAK Van Benthem et al. 1986. *Essays in logical semantics*. Springer.

Ilker Yildirim, Judith Degen, Michael K. Tanenhaus, and T. Florian Jaeger. 2013. Linguistic variability and adaptation in quantifier meanings. In *Proceedings of the 35th Annual Meeting of the Cognitive Science Society*.