Evaluation of a refinement algorithm for the generation of referring expressions

Luciana Benotti and Romina Altamirano

LIIS Team, Universidad Nacional de Córdoba, Argentina
benotti@famaf.unc.edu.ar, ialtamir@famaf.unc.edu.ar,
http://cs.famaf.unc.edu.ar/~luciana/,http://cs.famaf.unc.edu.ar/~romina/

Abstract. In this paper we describe and evaluate an algorithm for generating referring expressions that uses linear regression for learning the probability of using certain properties to describe an object in a given scene. The algorithm we present is an extension of a refinement algorithm modified to take probabilities learnt from corpora into account. As a result, the algorithm is able not only to generate correct referring expressions that uniquely identify the referents but it also generates referring expressions that are considered equal or better than those generated by humans in 92% of the cases by a human judge. We classify and give examples of the referring expressions that humans prefer, and indicate the potential impact of our work for theories of the egocentric use of language.

1 Introduction

A *referring expression* (RE) is an expression that unequivocally identifies the intended target to the interlocutor, from a set of possible distractors, in a given situation. For example, if we intend to identify a certain animal d from a picture of pets, the expression "the dog" will be an RE if d is the only dog in the picture, and if we are confident that our interlocutor will identify d as a dog.

The generation of referring expressions (GRE) is a key task of most natural language generation (NLG) systems [18]. Depending on the information available to the NLG system, certain objects might not be associated with an identifier which can be easily recognized by the user. In those cases, the system will have to generate a, possibly complex, description that contains enough information so that the interlocutor will be able to identify the intended referent. The generation of referring expressions is a well developed field in automated natural language generation building upon GRE foundational work [21, 6, 7].

Low complexity algorithms for the generation of REs have been proposed [3, 2]. These algorithms are based on variations of the partition refinement algorithms of [16]. The information provided by a given scene is interpreted as a relational model whose objects are classified into sets that fit the same description. This classification is successively *refined* till the target is the only element fitting the description of its class. The existence of an RE depends on the information available in the input scene, and on the expressive power of the formal language used to describe elements of the different classes in the refinement.

Existing GRE algorithms can effectively compute REs for all individuals in the domain, at the same time. The algorithms always terminate returning a formula of the formal language chosen that uniquely describes the target. However, GRE algorithms require a ranking of the properties that are to be used in the referring expressions, and the naturalness of the generated REs strongly depends on this ranking. [1] show that a refinement algorithm using the description language $\mathcal{EL}$ as formal language is capable of generating 75% of the REs present in the dataset described in [20]. In this paper we perform a human evaluation of the REs generated by this algorithm on two new corpora and show that even when the generated REs do not coincide with those found in corpora, people actually prefer the REs generated by the system in 92% of the cases.

The rest of the paper is structured as follows. In Section 2 we introduce the technical details of the refinement algorithm and explain how it uses the ranking of properties. In this section, we assume that this list is provided as input to the algorithm. In Section 3, we show how to estimate the probability of use of a property from corpora in order to obtain the ranking of properties. Given corpora consisting of pairs (scene, target) together with the REs used to describe the target in each case, we propose a method to compute the probability of use of each property for each scene, and use a machine learning approach to generalize this approach to new targets and scenes not appearing in the corpora. Section 4 presents an automatic evaluation and a human evaluation of the generated REs. In Section 5 we discuss related work and analyze the structure of the refinement algorithm in relation to the work of [12], on the egocentric basis of language generation.

2 The referring expression generation algorithm

Refinement algorithms for GRE are based on the following basic idea: given a scene S, the objects appearing in S are successively classified according to their properties into finer and finer classes. A description (in some formal language $\mathcal{L}$) of each class is computed every time a class is refined. The procedure always stops when the set of classes stabilizes, i.e., no further refinement is possible with the information available in the scene[1]. If the target element is in a singleton class, then the formal description of that class is a referring expression; otherwise the target cannot be unequivocally described (in $\mathcal{L}$).

We present a modification of the algorithm in [3] where the fixed order of properties in the input scene is replaced by a finite probability distribution. The resulting algorithm (see Figure 3) is now non-deterministic: two runs of the algorithm with the same input might result in different REs for objects in the scene. The input to the algorithm will be a relational model $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$, where Δ is the non-empty domain of objects in the scene, and $\|\cdot\|$ is an interpretation function that assigns to all properties in the scene its intended extension. For example, the scene shown in Figure 1 could be represented by the model $\mathcal{M} =$

[1] Of course, if we are only interested in a referring expression for a given target we can stop the procedure as soon as the target is the only element of some of the classes.

$\langle \Delta, \|\cdot\| \rangle$ shown in Figure 2. In Figure 2, $\Delta = \{e_1, \ldots, e_7\}$, and for example the extension of blue is $\|blue\| = \{e_5, e_6, e_7\}$ because 3 objects are blue in the scene. In the Figure, xn indicates that the object is in position n with regard to its x-dimension in the grid and yn is interpreted similarly.

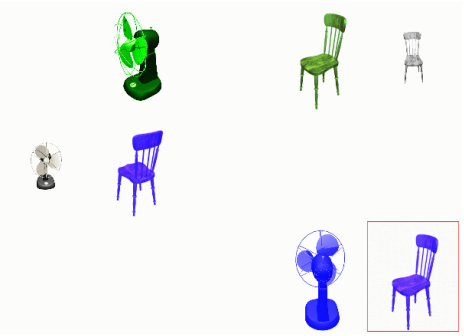

Fig. 1. Scene, target *blue chair facing left*

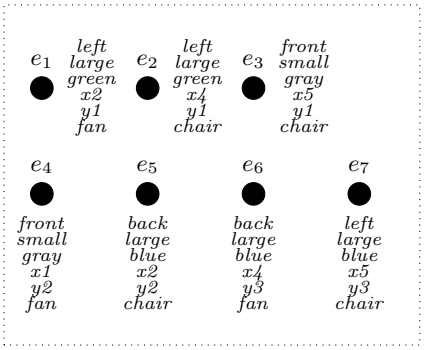

Fig. 2. The scene as a relational model

On termination, the algorithm computes what are called the $\mathcal{L}$-similarity classes of the input model $\mathcal{M}$. Intuitively, if two elements in the model belong to the same $\mathcal{L}$-similarity class, then $\mathcal{L}$ is not expressive enough to tell them apart (i.e., no formula in $\mathcal{L}$ can distinguish them). All the objects in Figure 1 are distinguishable, but if, for instance, color and position are not considered then e_2 and e_7 are indistinguishable and, hence, will remain in the same similarity class when the algorithm terminates.

The algorithm we discuss uses formulas of the $\mathcal{EL}$ description logic language [5] to describe refinement classes[2]. For a detailed description of $\mathcal{EL}$, we refer to [5]. The interpretation of the $\mathcal{EL}$ formula $\exists green.\top$ is the set of all the green elements of the model. In Figure 1, $\|\exists green.\top\| = \{e_1, e_2\}$. The interpretation of $\psi \sqcap \varphi$ is the set of all elements that satisfy ψ and φ. In Figure 1, $\|\exists green.\top \sqcap \exists chair.\top\| = \{e_2\}$.

Now that we have an intuitive understanding of $\mathcal{EL}$, we are ready to describe Algorithms 1 and 2.

Algorithm 1 takes as input a model and a list Rs of pairs (R,R.p_{use}) that links each relation R $\in$ REL, the set of all relation symbols in the model[3], to some probability of use R.p_{use}. For example, *green* and *large* are relations in the model of Figure 2. The set RE contains the formal description of the refinement classes and it is initialized by the most general description $\top$. The formula $\top$ can be intuitively understood as the referring expression *thing* or *thingummy*. For each R, we first compute R.rnd_{use}, a random number in [0,1]. If R.$\mathsf{rnd}_{use} \leq$ R.p_{use} then R is used to refine the set of classes. The value of R.p_{use} will be incremented

[2] Notice, though, that the particular formal language used is independent of the main algorithm, and different add$_{\mathcal{L}}$(R,φ,RE) functions can be used depending on the language involved.

[3] We represent each unary relation R as binary, hence $\|\exists R.\top\|$ is the set of all elements in the model that have the property R

Algorithm 1: Computing $\mathcal{L}$-similarity classes

Input: A model $\mathcal{M}$ and a list Rs $\in (\mathsf{REL} \times [0,1])^*$ of relation symbols with their p_{use} values, ordered by p_{use}

Output: A set of formulas RE such that $\{\|\varphi\| \mid \varphi \in \mathsf{RE}\}$ is the set of $\mathcal{L}$-similarity classes of $\mathcal{M}$

```
RE ← {⊤}   // the most general description ⊤ applies to all elements in the scene
for (R,R.p_use) ∈ Rs do
    R.rnd_use = Random(0,1)      // R.rnd_use is the probability of using R
    R.inc_use = (1 − R.p_use) / MaxIterations
repeat
    while ∃(φ ∈ RE).(#‖φ‖ > 1) do  // while some class has at least two elements
        RE' ← RE                    // make a copy for future comparison
        for (R, R.p_use) ∈ Rs do
            if R.rnd_use ≤ R.p_use then  // R will be used in the expression
                for φ ∈ RE do add_EL(R, φ, RE)  // refine classes using R;
            if RE ≠ RE' then          // the classification has changed
                exit          // exit for-loop to try again highest R.p_use
        if RE = RE' then              // the classification has stabilized
            exit                      // exit while-loop to increase R.p_use
    for (R,R.p_use) ∈ Rs do R.p_use ← R.p_use + R.inc_use     // increase R.p_use ;
until ∀((R,R.p_use) ∈ Rs).(R.p_use ≥ 1)      // R.p_use are incremented until 1;
```

Algorithm 2: $\mathrm{add}_{\mathcal{EL}}(\mathrm{R}, \varphi, \mathsf{RE})$

```
if FirstLoop? then                          // are we in the first loop?
    Informative ← TRUE                      // allow overspecification
else Informative ← ‖ψ ⊓ ∃R.φ‖ ≠ ‖ψ‖;     // informative: smaller than the original?
for ψ ∈ RE with #‖ψ‖ > 1 do
    if ψ ⊓ ∃R.φ is not subsumed in RE and      // non-redundant: can't be obtained from RE?
       ‖ψ ⊓ ∃R.φ‖ ≠ ∅ and                      // non-trivial: has elements?
       Informative then
        add ψ ⊓ ∃R.φ to RE    // add the new class to the classification
        remove subsumed formulas from RE     // remove redundant classes
```

Fig. 3. Refinement algorithm with probabilities for the $\mathcal{EL}$-language

by $\mathrm{R}.\mathsf{inc}_{use}$ in each main loop, to ensure that all relations are, at some point, considered by the algorithm. This ensures that a referring expression will be found if it exists; but gives higher probability to expressions using relations with a high $\mathrm{R}.\mathsf{p}_{use}$. While RE contains descriptions that can be refined (i.e., classes with at least two elements) the refinement function $\mathrm{add}_{\mathcal{L}}(\mathrm{R},\varphi,\mathsf{RE})$ is called successively with each relation in Rs. If the model contains binary relations

between its elements, a change in one of the classes, can trigger changes in others. For that reason, if RE changes, we exit the **for** loop to start again with the relations of higher R.p_{use}. If after trying to refine the set with all relations in Rs, the set RE has not changed, then we have reached a stable state (i.e., the classes described in RE cannot be further refined with the current R.p_{use} values). We will then increment all the R.p_{use} values and start the procedure again.

Algorithm 2 behaves as follows. The **for** loop refines each description in RE using the relation R and the other descriptions already in RE, under certain conditions. The new description should be *non-redundant* (it cannot be obtained from classes already in RE), *non-trivial* (it is not empty), and *informative* (it does not coincide with the original class). If these conditions are met, the new description is added to RE, and redundant descriptions created by the new description are eliminated. The **if** statement at the beginning of Algorithm 2 disregards the informativity test during the first loop of the algorithm allowing overspecification; without this condition the algorithm would generate minimal REs. For example, a minimal RE for e_2 is "the green chair" while an overspecified RE for this element is "the green chair in the top row".

3 Learning to describe new objects from corpora

In the previous section we presented an algorithm that assumes that each relation R used in a referring expression has a known probability of use R.p_{use}. Intuitively, the R.p_{use} is the probability of using relation R to describe the target. In Tables 1 and 2 we show the probabilities of use that we are able to learn from corpora and to apply to the models of Figures 1 and 4. In Figure 1, the probability of using *blue* to describe the target is higher than the probability of using *facing left*, although both are properties of the target.

The probability of using *green* is not zero because a green object may be used in a relational description of the target (for example, "the blue chair far from the green fan").

In this section, we describe how to calculate these probabilities from corpora. The general set up is the following: we assume available a corpus of REs associated to different scenes that are prototypical of the domain in which the GRE algorithm has to operate; we call this the training data. We then show how to generalize these values to other scenes in the domain, using a machine learning algorithm. We exemplify the methodology using the TUNA corpus.

The TUNA Corpus [10] is a set of human-produced referring expressions (REs) for entities in visual domains of pictures of furniture and people as exemplified in Figures 1 and 4. The corpus was collected during an online elicitation experiment in which subjects typed descriptions of a target single referent or pair of referents. In each picture there were 5 or 6 other objects. In the experiment, the participation was not controlled, but there was a main condition manipulated the +/-LOC: in +LOC condition, participants were told that they could refer to entities using any of their properties (including their location on the screen). In the -LOC condition, they were discouraged from doing so, though not prevented.

Top 10 relations in Figure 1	learned p_{use}
chair	0.94
blue	0.89
y3	0.29
x5	0.27
left	0.25
large	0.21
green	0.05
small	0.05
back	0.02
y1	0.02

Table 1. Probabilities of use learned from corpora and instantiated for Figure 1

Top 10 relations in Figure 4	learned p_{use}
person	0.79
hasGlasses	0.71
y2	0.20
x5	0.18
hasHair	0.13
hairDark	0.13
hairLight	0.11
ageOld	0.05
y3	0.03
x2	0.02

Table 2. Probabilities of use learned from corpora and instantiated for Figure 4

The attributes for each entity include properties such as an object's color or a person's characteristic such as having dark hair. In this paper we will use the singular part of the TUNA corpus. The corpus contains 780 singular referring expressions divided into 80% training data, 20% test.

In order to collect the corpus, each participant in the elicitation experiment carried out 38 trials, 20 furniture descriptions and 18 people descriptions. For each word in the corpus we train a machine learning model that computes a function of its p_{use}. When this function is instantiated with a set of domain independent features that we define below.

Fig. 4. Scene used during the collection of the TUNA corpus. The referring expression collected has to distinguish the target from the rest of the people. For this scene, the RE was *the man with glasses*

.

To clarify the computation of R.p_{use} in the training data and the model $\mathcal{M}$ associated to each scene we list the required steps in detail, and discuss how we carried them out in the TUNA corpus:

1. Tokenize the referring expressions and call the set of tokens T. In particular, multi-word expressions like "in the top row" should be matched to a single token like *y1*.
2. Replace hyperonyms from T. E.g., if both *man* and *picture* appear in T, delete *picture*.
3. If the set of tokens obtained in the previous steps contains synonyms normalize them to a representative in the synonym class, and call the resulting set REL; it will be the signature of the model $\mathcal{M}$ used by the algorithm. E.g., the tokens *man* and *guy* are both represented by the token *man*.
4. For each scene, define $\mathcal{M}$ such that the interpretation $\|\cdot\|$ ensures that all REs in the corpus are REs. E.g., the $\mathcal{EL}$ formula $\exists \mathit{left}.\top \sqcap \exists \mathit{blue}.\top \sqcap \exists \mathit{chair}.\top$, which represents the RE "the blue chair facing left" found in the corpus for the scene in Figure 1, is a RE for the target in the model $\mathcal{M}$ depicted in Figure 2.
5. For each $\mathrm{R} \in \mathsf{REL}$ we assign 1 to $\mathrm{R}.\mathsf{p}_{use}$ if R occurs in the RE, we assign 0 otherwise. In case that the corpus has more than one RE per scene we calculate the proportion of appearance of each property.

The learning was done with the machine learning toolkit WEKA [11], learning on the training data of the TUNA corpus. We use linear regression to learn the function of p_{use} for each relation in the signature. For a given scene in the test set, we replace the variables of the obtained function by the values of the features in the scene that we want to describe. We use simple features to obtain the function, all the features can be extracted automatically from the relational model and are listed in Table 3.

target-has	whether the target element has the property
location-has	whether the RE may use the location of the target in the figure
discrimination	1 / the number of objects in the model that have the property
p_{use}	probability of using the property to describe the target

Table 3. Features used for learning the p_{use} for each token in the signature of the scenes of the TUNA corpus

Our feature set is intentionally simplistic in order for it to be domain independent. As a result there are some complex relations between characteristics of the scenes that it is not able to capture.

Starting from the scene in Figure 1 the resulting signature and their associated p_{use} are listed in the Table 1 and for the Figure 4 in Table 2. Notice that even though the TUNA corpus contains only one RE per scene the p_{use} values represent the proportion of use of each property as learned using linear regression.

Notice that the values $\mathrm{R}.\mathsf{p}_{use}$ obtained in this way should be interpreted as the probability of using R to describe the target in model $\mathcal{M}$, and we could argue that they are correlated to the *saliency* of R in the model.

Using linear regression we are able to learn interesting characteristics of the domain. To start with, it learns known facts such that the saliency of a color

depends strongly on whether the target object is of that color, and it does not depend on its discrimination power in the model. Moreover, it learns that size relations (e.g., large and small) are used more frequently when it has a higher discriminative power which confirms a previous finding reported in [20]. Finally, it is able to learn that the orientation properties (e.g., facing left and facing right) are used as a last resource, when it is necessary to identify the target uniquely.

4 Evaluation

In this section we present two different evaluations we performed on our algorithm. Section 4.1 describes an evaluation with respect to the state of the art algorithm GRAPH [13]. GRAPH was the top performer in both editions of the ASGRE, shared task [10]. Due to the limitations of the automatic metrics, in Section 4.2 we perform a human evaluation in which we ask human subjects to compare the output produced by our algorithm to expressions produced by humans.

4.1 Automatic evaluation

In this section we present the comparison of our algorithm to the state of the art algorithm GRAPH introduced above. The GRAPH algorithm is a deterministic algorithm and hence produces the same referring expression when run with the same target and model. Our algorithm is non deterministic, it may give a different referring expression each time it is run. In order to compare them we run our algorithm k times and we make a ranking of the top 20 produced referring expressions ordered by the frequency they were produced. We use the test part of the TUNA corpus to compare algorithm to the GRAPH algorithm whose results on this dataset are described in [13] and reproduced in the Table 4.

The GRAPH algorithm defines the generation of referring expressions as a graph search problem, which outputs the cheapest distinguishing graph (if one exists) given a particular cost function. We compare to this algorithm using the metrics accuracy, Dice and MASI. Accuracy is defined as the percentage of exact matches between each RE produced by a human and the RE produced by the system for the same scene.

Dice coefficient is a set comparison metric, ranging between 0 and 1, where 1 indicates a perfect match between sets. For two attribute sets A and B, Dice is computed as follows:

$$Dice(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}$$

The MASI score [17] is an adaptation of the Jaccard coefficient which biases it in favor of similarity where one set is a subset of the other. Like Dice, it ranges between 0 and 1, where 1 indicates a perfect match. It is computed as follows:

$\text{MASI}(A, B) = \delta \times \frac{|A \cap B|}{|A \cup B|}$

where δ is a monotonicity coefficient defined as follows:

$$\delta = \begin{cases} 0 \ if A \cap B = \emptyset \\ 1 \ if A = B \\ \frac{2}{3} \ if A \subset B \ or \ B \subset A \\ \frac{1}{3} \ otherwise \end{cases} \quad (1)$$

Intuitively, this means that those system-produced descriptions are preferred which do not include attributes that are omitted by a human.

In Table 4 we show the automatic metrics and compare the performance of our system with the GRAPH system for the first RE in the ranking and the first 20 REs in the ranking.

	Dice	MASI	ACCURACY
GRAPH system, Furniture domain	.80	.59	.48
GRAPH system, People domain	.72	.48	.28
Our system, Furniture domain (top 1)	.80	.60	.47
Our system, People domain (top 1)	.65	.37	.19
Our system, Furniture domain (top 20)	.87	.75	.65
Our system, People domain (top 20)	.81	.68	.60

Table 4. Comparison of the GRAPH algorithm and our system. We consider the 3 automatic metrics for the top 1 and the top 20 REs produced by our algorithm.

Accuracy, Dice and MASI assess humanlikeness with respect to a corpus of human referring expressions. In the Figure 5 the accuracy for our system and the GRAPH system is compared. The left GRAPH corresponds to the furniture domain and the right GRAPH corresponds to the people domain. We can see that taking the top 1 RE our system accuracy is lower than GRAPH performance for the people domain. However, if we consider the top 20 REs that our algorithm is able to produce we can see that the accuracy for both domains gets higher than 60%. This shows that our algorithm is able to generate REs that are more similar to those produced by humans than the GRAPH algorithm, although these REs are not ranked first.

Another result that we can observe is that the people domain accuracy is much lower for the top 1 RE than for the furniture domain (19 vs 47), but the accuracy stabilizes when REs lower in our ranking are considered. This may be explained by the fact that the training set for the people domain is smaller and less balanced and hence, the probabilities of use inferred do not generalize as well as in the furniture domain.

4.2 Human evaluation

We asked two native speaker judges of English to evaluate our referring expressions via an experiment on the web. The authors of the paper did not participate

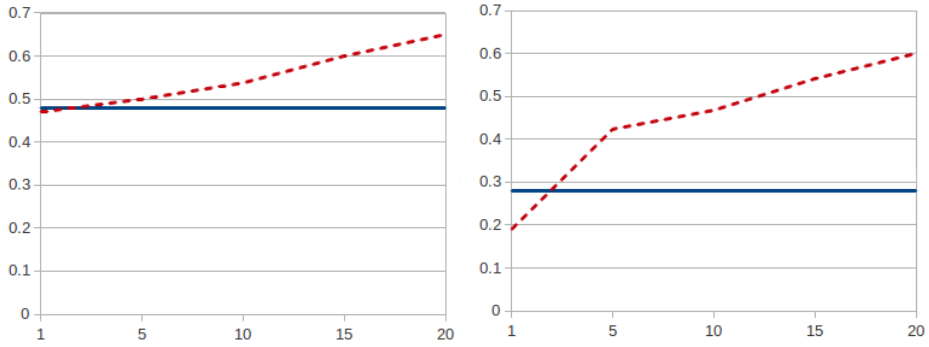

Fig. 5. Comparison of the accuracy of the GRAPH algorithm and our system. The x axis indicates that the accuracy was calculated considering the x first REs in the ranking. The y axis indicates the accuracy. Our system is depicted as a dotted line and the GRAPH system as a continuous line.

during the evaluation. The judges could register to the evaluation system so that they did not have to complete it in one go, the could come back to it later. During the evaluation we showed each judge the scenes and two randomly ordered REs. One RE corresponded to the RE present in the corpus and produced by a person and the other RE corresponded to the top 1 RE produced by our system. We asked the judges to select the RE that would be more useful to identify the target in the scene. That is, to select it from among the other objects in the stimulus pictures.

Our goal is to show that even if the RE generated by our algorithm does not coincide with the RE produced by a human in the corpus collection, it can be judged as good or even better than the REs generated by humans.

In Table 5 we show the results from the human evaluation experiment. The REs produced by the system were considered equal or better by both judges in 60 % of the cases and, by at least one judge in 92% of the cases.

	Furniture domain	People domain	Weighted mean
system equal to human	.46	.19	.33
system better by 2 judges	.29	.24	.27
system better by 1 or 2 judges	.51	.68	.59
system worse by 2 judges	.03	.13	.08
system equal or better by 2 judges	.75	.43	.60
system equal or better by 1 judge	.97	.87	.92

Table 5. Percentage of system versus human selected choices

Below, we illustrate the evaluation experiment by showing examples of cases in which the system expression was considered better by both judges, by only one judge or by neither of them.

Figure 6 illustrates a case in which the human generated an underspecified RE while the system produced an RE which unequivocally identifies the target. The RE generated by the system for this figure is "small blue fan" while the RE produced by the human is "blue fan". The human RE fails to uniquely identify the target and is then not preferred by the human judges. Humans are known for producing underspecified REs which may be due to cognitive limitations for not being able to consider the whole referential context at the same time. Our algorithm is able to consider the whole referential context and combine this ability with the probability of use of the REs learned from humans.

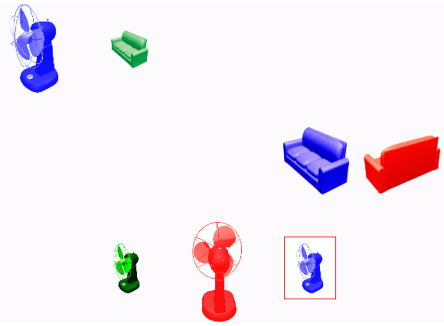

Fig. 6. Scene used during the collection of the TUNA corpus. The human RE *blue fan*, and the system *small blue fan*. Judges prefer the system generated.

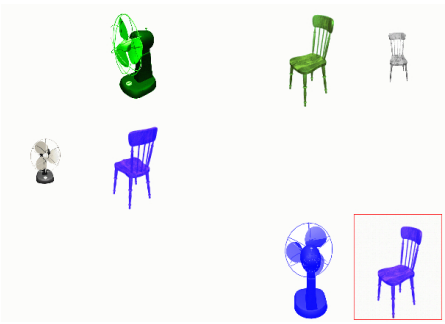

Fig. 7. Scene used during the collection of the TUNA corpus. The human RE was *blue frontal chair*, and the system *the blue chair in the bottom*. Both human judges prefer the system generated RE.

In Figure 7 the human RE was "blue frontal chair", and the system RE was "the blue chair in the bottom"; both judges selected the system RE. This case can be explained by the fact that, in this domain, the property "bottom" helps more during the identification than the property "frontal" because it concentrates the attention of the interpreter in the lower part of the scene. Our system learns this fact by learning a higher value of p_{use} for "bottom" than for "frontal" from the training data.

Figure 8 is an example for which both judges preferred the human expression. The human RE was "the man with black hair", and the system's "the man wearing glasses in the fourth column". This example makes evident the fact that, in the people domain some properties are more salient in some images than in others because of different shades of colors. Gradable properties such as this ones (in contrast to absolute properties) are still an open problem for GRE algorithms.

Figure 9 illustrates a case in which the system RE was more overspecified than the human RE; the system included "wearing glasses" while the human did not. In this case one human subject preferred the system RE and the other the

Fig. 8. Scene used during the collection of the TUNA corpus. The human RE was *the man with black hair*, and the system *the man wearing glasses in the fourth column*. Judges prefer the human RE.

Fig. 9. Scene used during the collection of the TUNA corpus. The human RE was *man with a beard*, and the system *man with a beard wearing glasses*. Judges did not agree in their preference.

human RE. The amount of overspecification is a subjective matter where human themselves disagree. Further evaluation where REs are actually used for a task would be interesting to investigate this issue.

5 Discussion and Conclusions

In this article we presented the evaluation of the algorithm presented in [3] extended to generate REs similar to those produced by humans. The modifications proposed are based on the observation that humans frequently overspecify their REs [8, 4]. We tested the proposed algorithm on the TUNA corpus and found that it is able to generate a large proportion of the overspecified REs found in the corpus without generating trivially redundant referring expressions. The expressions generated are preferred by (one or more) human judges 92% of the time for the TUNA corpus.

Different algorithms for the generation of overspecified and distinguishing referring expressions have been proposed in recent years (see, e.g., [14, 19]). In this paper we compare ourselves to the Graph algorithm [13] wich has been shown to achieve better accuracy than algorithms describe in [14, 19] in the TUNA shared task [10].

An interestesting outcome of our work is that it makes evident the relationship between overspecification and the saliency of properties in the context os a scene.

As we described in Section 2 the generation of overspecified REs is performed in two steps. In the first iteration, the probability of including a property in the RE depends only on its p_{use}. We believe our definition of p_{use} is intended to captures the saliency of the properties for different scenes and targets. The p_{use} of a property changes according to the scene as we discussed in Section 3. This is in contrast with previous work where the saliency of a property is constant in

a domain. In the first iteration, if the p_{use} is high, that is, if the property is very salient, it does not matter whether the property eliminates any distractor, it will probably be used anyway. After all properties had a chance of being included in this way, if the resulting RE is not distinguishing, then the algorithm enters a second phase in which it makes sure that the RE identifies the target uniquely.

Our two-step algorithm is inspired by the work of [12] on egocentrism and natural language production. Keysar et al. put forwards the proposal that when producing language, considering the hearers point of view is not done from the outset but it is rather an afterthought [12]. They argue that adult speakers produce REs egocentrically, just like children do, but then adjust the REs so that the addressee is able to identify the target unequivocally. The egocentric step is a heuristic process based in a model of saliency of the scene that contains the target. As a result, the REs that include salient properties are preferred by our algorithm even if such properties are not necessary to identify the target univocally. Keysar et al. argue that the reason for the generate-and-adjust procedure may have to do with information processing limitations of the mind: if the heuristic that guides the egocentric phase is well tunned, it succeeds with a suitable RE in most cases and seldom requires adjustments. Interestingly, we observe a similar behavior with our algorithm: when p_{use} values learn from the domain are used, the algorithm is not only much more accurate but also much faster.

As future work we plan to evaluate our algorithm to generate referring expressions inside discourse as required by domains like those provided by Open Domain Folksonimies [15]. We also plan to explore corpora obtain from interaction, such as the GIVE Corpus [9] where it is common to observe multi shot REs. Under time pressure subjects will first produce an underspecified expression that includes salient properties of the target (e.g., "the red button"). And then, in a following utterance, they add additional properties (e.g., "to the left of the lamp") to make the expression a proper RE identifying the target uniquely.

References

1. Altamirano, R., Areces, C., Benotti, L.: Probabilistic refinement algorithms for the generation of referring expressions. In: Proceedings of the 24th International Conference on Computational Linguistics. pp. 53–62 (2012)
2. Areces, C., Figueira, S., Gorín, D.: Using logic in the generation of referring expressions. In: Pogodalla, S., Prost, J. (eds.) Proceedings of the 6th International Conference on Logical Aspects of Computational Linguistics (LACL 2011). Lecture Notes in Computer Science, vol. 6736, pp. 17–32. Springer (2011)
3. Areces, C., Koller, A., Striegnitz, K.: Referring expressions as formulas of description logic. In: Proceedings of the 5th International Natural Language Generation Conference (INLG'08). pp. 42–49. Association for Computational Linguistics, Morristown, NJ, USA (2008)
4. Arts, A., Maes, A., Noordman, L., Jansen, C.: Overspecification facilitates object identification. Journal of Pragmatics 43(1), 361–374 (2011)

5. Baader, F., McGuiness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, implementation and applications. Cambridge University Press (2003)
6. Dale, R.: Cooking up referring expressions. In: Proceedings of the 27th annual meeting on Association for Computational Linguistics. pp. 68–75 (1989)
7. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. Cognitive Science 19(2), 233–263 (1995)
8. Engelhardt, P., Bailey, K., Ferreira, F.: Do speakers and listeners observe the gricean maxim of quantity? Journal of Memory and Language 54(4), 554–573 (2006)
9. Gargett, A., Garoufi, K., Koller, A., Striegnitz, K.: The GIVE-2 corpus of giving instructions in virtual environments. In: Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC). Malta (2010)
10. Gatt, A., Belz, A., Kow, E.: The TUNA challenge 2008: Overview and evaluation results. In: Proceedings of the 5th International Conference on Natural Language Generation. pp. 198–206. Association for Computational Linguistics (2008)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (Nov 2009)
12. Keysar, B., Barr, D.J., Horton, W.S.: The Egocentric Basis of Language Use. Current Directions in Psychological Science 7(2), 46–49 (Apr 1998)
13. Krahmer, E.J., Theune, M., Viethen, J., Hendrickx, I.: Graph: The costs of redundancy in referring expressions. In: Proceedings of the 5th International Natural Language Generation Conference, Salt Fork, Ohio, USA. pp. 227–229. The Association for Computational Linguistics, USA (June 2008)
14. de Lucena, D.J., Paraboni, I.: USP-EACH frequency-based greedy attribute selection for referring expressions generation. In: Proceedings of the 5th International Conference on Natural Language Generation (INLG 2008). pp. 219–220. Association for Computational Linguistics (2008)
15. Pacheco, F., Duboue, P., Domínguez, M.: On the feasibility of open domain referring expression generation using large scale folksonomies. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 641–645. Association for Computational Linguistics, Montréal, Canada (June 2012)
16. Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM Journal on Computing 16(6), 973–989 (1987)
17. Passonneau, R.: Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In: In Proceedings of the International Conference on Language Resources and Evaluation (LREC (2006)
18. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, Cambridge (2000)
19. Ruud, K., Emiel, K., Mariët, T.: Learning preferences for referring expression generation: Effects of domain, language and algorithm. In: INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference. pp. 3–11. Association for Computational Linguistics, Utica, IL (May 2012)
20. Viethen, H.A.E.: The Generation of Natural Descriptions: Corpus-Based Investigations of Referring Expressions in Visual Domains. Ph.D. thesis, Macquarie University, Sydney, Australia (2011)
21. Winograd, T.: Understanding natural language. Cognitive Psychology 3(1), 1–191 (1972)