Statistical NLG for Generating the Content and Form of Referring Expressions

Xiao Li and **Kees van Deemter** and **Chenghua Lin**
Computing Science department
University of Aberdeen
Aberdeen, UK
{xiao.li, k.vdeemter, chenghua.lin}@abdn.ac.uk

Abstract

This paper argues that a new generic approach to statistical NLG can be made to perform Referring Expression Generation (REG) successfully. The model does not only select attributes and values for referring to a target referent, but also performs Linguistic Realisation, generating an actual Noun Phrase. Our evaluations suggest that the attribute selection aspect of the algorithm exceeds classic REG algorithms, while the Noun Phrases generated are as similar to those in a previously developed corpus as were Noun Phrases produced by a new set of human speakers.

1 Introduction

Referring Expressions Generation (REG) is a subtask of Natural Language Generation (NLG) that decides how to distinguish a target referent from its distractors, as when we say "the sofa", "the red sofa", and so on, to distinguish the referent from other furniture. Most current REG algorithms are rule-based (Gatt and Krahmer, 2018), though Machine Learning is also starting to be used (e.g. Di Fabbrizio et al., 2008).

REG is usually treated as an independent stage or component of NLG pipelines (e.g. Reiter and Dale, 2000; Reiter, 2007; Gatt and Krahmer, 2018; Di Fabbrizio et al., 2008). The present paper changes the relationship between NLG and REG: it regards REG as a special case of usual NLG, and proposes a vector-based algorithm to transform REG tasks into a generic NLG tasks. The paper adopts the NLG algorithm of our previous work (which is also vector-based; Li, 2019), but certain adaptations needed to be made to allow the algorithm to perform the traditional REG attribute selection task.

Our REG algorithm produces referring expressions (REs) by learning from a data-text corpus of REs. In a nutshell, the algorithm splits the textual expressions in the training corpus into small spans according to their meaning, and reassembles these spans into new expressions when it refers to a referent. We evaluate the performance of the REG function on the Tuna corpus (Gatt et al., 2007) against 3 strong baselines. Experimental results show that our algorithm outperforms the baselines in terms of Dice scores. An additional experiment also shows positive results for our algorithm on an experts-based evaluation based on a BLEU-based comparison between algorithm-generated and human-produced referring expressions.

2 Related Work

The main REG algorithms have often focussed on the semantic core of the REG task, which is to select semantic attributes for a referring expression (e.g., `sofa`, `red`), disregarding the expression of these attributes in words (e.g., "the red sofa") (Krahmer and Van Deemter, 2012). The term REG is sometimes restricted to "one-shot" references, where it is the task of one single NP to identify the referent. We will use the term REG in this restricted sense. Consequently, linguistic context is irrelevant, so pronouns and other anaphoric NPs (as in the GREC challenge, for instance (Gatt et al., 2009)) will not be taken into account.

Early REG approaches sought to find a *minimum* set of attributes that jointly single out the referent, this is called the Full Brevity (FB) approach (Dale, 1989), or to "greedily" add maximally discriminatory attributes one by one, that is, adding attributes one at a time, choosing always the one that removes the largest number of distractors, until the referent has been uniquely identified; this is called the Greedy algorithm (GR).

The approach that is often thought to be most suitable for relatively simple referential situations (cf. §6 below), known as the Incremental Algorithm (IA), resembles GR to the extent that it adds attributes one by one until the referent has been singled out. However, the order in which the IA selects attributes is not based on their discriminatory power (as was the case with GR), but on their place in what is known as the Preference Order (PO). The PO is a list that works like an oracle that tells us in what order Attributes should be selected and, although efforts have been made to determine the PO on independent grounds, in practice it is nontrivial to find the best PO for a given REG task. Thus each PO defines another IA. Therefore, when we evaluate our own REG approach in §4.3, we will not only compare it with FB and GR, but with a number of different IAs, with different POs.

Statistical methods have come to REG relatively late. The hybrid approach of Di Fabbrizio et al. (2008) used statistical methods to determine the PO of the IA. The Bayesian approach of Frank and Goodman (2012) went further, but although it has some attractive features, it does not yet perform at the same level as IA. Other statistical approaches have focussed specifically on the logical structure of complex REs (FitzGerald et al., 2013) and on collaborative aspects of referring (Garoufi and Koller, 2014), among other issues.

REG algorithms have typically focussed on Content Selection (i.e., selection of semantic attributes). For example, when the usefulness of REs for readers was addressed in the *Shared Task Evaluation Challenges* (STECs; Gatt and Belz, 2010), the sets of properties produced by each of the algorithms submitted to the STECs were converted into actual Noun Phrases (by one and the same simple Linguistic Realisation algorithm) before they were shown to readers.

3 Summary of the Text-Reassembling Generation Model

We are developing a new approach to NLG (Li, 2019), which we call the *Text-Reassembling Generation* (TRG) model. Earlier experiments with this method have focussed on the SUMTIME corpus (Sripada et al., 2002), and more specifically on Lexical Choice and the generation of SUMTIME-style sentences such as "MAINLY W-NW 10 OR LESS" (a brief weather prediction about the strength and direction of wind). In the present paper, we show how this approach can be adapted to perform the REG task (a task not previously considered in this work). Here we sketch the outlines of TRG; the next section applies these ideas to REG.

Table 1: A simplified training corpus for training the Vector-Based Approach to NLG. Wind speed (`ws`) are expressed in Knots and wind direction (`dir`) is presented by Compass points.

Text	Data
W 10-12	{`ws=`*`10`*`,dir=`*`265`*}
WS 22-24	{`ws=`*`22`*`,dir=`*`130`*}
MAINLY 10 OR LESS	{`ws=`*`9`*`,dir=`*`10`*}
...	...

TRG uses a generation strategy that we call "splitting-and-reassembling". It splits the training sentences into text fragments (i.e. the strings consist of words, numbers, punctuations, and so on) during training; then, it reuses (reassembles) the fragments to generate new sentences. The training process aims to extract sentence fragments. From a training corpus, the approach learns what fragment express what non-linguistic data by inspecting what non-linguistic data most likely co-occurs with what fragment.

Vector-based Knowledge Representation (KR) is often used in deep learning-based and other Machine Learning approaches (Ramachandram and Taylor, 2017). TRG adopts *attribute-value* pairs for representing non-linguistic data (see §4.1 for details), and the attribute-value pairs are further represented by vectors. Table 1 presents a simplified training corpus of SUMTIME sentences about wind. For example, the fragment "10-12" co-occurs with `ws=`*`10`* (i.e., the first data record). If this pattern was observed frequently in the training set, our model can learn that the fragment "10-12" expresses `ws=`*`10`*.

Another part of our approach is schema extraction, in which each text fragment is replaced by a placeholder that leaves out everything except the type of attribute expressed. After the replacement, the text becomes a sequence of placeholders, and we call the sequence the *schema*. For example, revisiting the example of Figure 1, the three corpus texts are transformed into two schemas:

W 10-12	⇒ [direction] [speed]
WS 22-24	⇒ [direction] [speed]

MAINLY 10 OR LESS ⇒ [speed]

The two schemas are:

schema 1: [direction] [speed]
schema 2: [speed]

where the placeholders are denoted by square brackets.

Generation is a two-step process: given a set of non-linguistic data as input, TRG firstly selects a schema. It then replaces each placeholder in the schema with a text fragment which expresses the non-linguistic data (i.e., the input). Note that this approach can generate texts which do not appear in the training corpus. Because TRG represents input data by vectors, both schema selection and text fragment selection are based on vector comparison between the input data vector and corpus data vectors. Unlike most of its predecessors, it is *purely* statistical in that it does not use a hand-crafted grammar or other hand-crafted rules.

4 The Vector-Based REG Algorithm

In this section we show how the TRG algorithm was made to apply to REG, which generates expressions that distinguish a target from its distractors. A RE typically expresses only a subset of the features of the target referent. For example, although in the Tuna corpus, furniture has 4 properties including `type`, `colour`, `size`, and `orientation`. Speakers referring to a large red frontal chair (in the presence of other large frontal chairs) may only say "the red chair", because this suffices to distinguish the referent from all the other objects in the domain (called the distractors). The choice of features tells us something about the referent, but also about the differences between the target(s) and distractors.

To generate an appropriate referring expression, information of both the target and the distractors needs to be considered. To bring REG within the scope of TRG, we therefore combine the features of the target referent with the differences between the target and distractors, treating this as a new group of features. Then we generate a RE as well as a description of the new group, and this description is the textual RE for the target. The resulting algorithm is called the Vector-Based approach to REG (VB-REG).

In the following sections, we first introduce how we represent a domain object (i.e., a target or a distractor) and the differences between the target and distractors (§4.1). We then discuss our strategy for generating REs for a target referent (§4.2 and §4.3). Finally, we explain how to generate an actual referential noun phrase (§4.4).

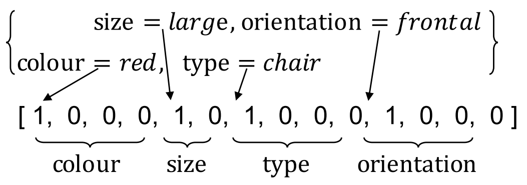

Figure 1: An example of vector representation for a domain object with the feature set {colour=*red*, size=*large*, type=*chair*, orientation=*frontal*}.

4.1 Representing a Reference Problem as a Fixed-Length Vector

We represent a target or a distractor with features which are represented as *attribute-value* pairs, e.g., colour=*red* and type=*chair*. Here colour and type are attributes and *red* and *chair* are the values. A chair whose colour is red, size is large, and orientation is frontal can be represented by the feature set: {colour=*red*, size=*large*, type=*chair*, orientation=*frontal*}.

In order to apply our model, we need to represent feature sets in a vector. Recall that there are in total 14 features in the Tuna corpus (Gatt et al., 2007); they are:

colour=*red*	type=*chair*
colour=*blue*	type=*desk*
colour=*green*	type=*fan*
colour=*grey*	type=*sofa*
orientation=*front*	size=*large*
orientation=*back*	size=*small*
orientation=*left*	
orientation=*right*	

We therefore use a 14-dimension vector to represent an arbitrary feature set, with each dimension corresponding to an attribute-value pair. Also, each dimension is a binary variable (i.e., 1 or 0), with 1 indicating that the corresponding feature appears in the feature set, and 0 otherwise. In this way, the feature set {colour=*red*, size=*large*, type=*chair*, orientation=*frontal*} of the expression "a large red frontal chair" can be converted to a vector $[1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0]$, that we

Table 2: An example of a simplified REG case from the Tuna corpus.

Target:	{colour=*grey*, size=*large*, ...}
Distractor 1:	{colour=*blue*, size=*small*, ...}
Distractor 2:	{colour=*red*, size=*large*, ...}
Distractor 3:	{colour=*blue*, size=*large*, ...}
Distractor 4:	{colour=*red*, size=*large*, ...}

call the target vector (namely $\mathbf{t}$). Two objects are indistinguishable if their vector representations are the same.

Since the words that are used for referring to an object depend not just on that object (i.e., the referent), but on other objects as well (i.e., the distractors), we model the difference between a target and its distractors with an additional vector, which we call the difference vector ($\mathbf{d}$). In contrast to the feature set vector, here, the difference vector has only 4 dimensions, with each dimension corresponding to one of the attribute types of the Tuna corpus, i.e., colour, size, type, and orientation. The value of the dimensions indicates the degree to which the target differs from the distractors for a corresponding attribute.

Let $attr$ be an attribute, P_{attr} the probability that the value of $attr$ of the target matches the same attribute value of any of the distractors.

$$P_{attr} = \frac{count(attr_{target} = attr_{distractor})}{count(distractor)} \quad (1)$$

We show how to compute the difference vector $\mathbf{d}$ with the example in Table 2.

In Table 2, the value of the colour attribute of the target is grey. However, no distractor has the same value for the colour attribute, meaning that $P_{colour} = 0$. Thus, we have

$$\mathbf{d}^{(colour)} = 1 - P_{colour} = 1 \quad (2)$$

This means that target's colour is very different from the distractors, i.e., an outlier among the colour of the distractors. For the size attribute, target's size feature (i.e., size=*large*) occurs three times among the four distractors. Thus, we have

$$P_{size} = \frac{3}{4}$$
$$\mathbf{d}^{(size)} = 1 - P_{size} = \frac{1}{4}$$

That indicates that the size of the target is unlikely to be an outlier to the distractors. Finally, the difference vector $\mathbf{d}$ is given as

$$\mathbf{d} = [\mathbf{d}^{(colour)}, \mathbf{d}^{(size)}, \mathbf{d}^{(type)}, \mathbf{d}^{(orien)}] = [1, \frac{1}{4}, ...]$$

As we described, the referring expression generation considers both the target information (presented by $\mathbf{t}$) and the differential (presented by $\mathbf{d}$) between the target and distractors. We join the two vectors as a big vector (namely *knowledge vector* or $\mathbf{k}$) for combining the two parts of information. In this case, $\mathbf{k}$ is:

$$\mathbf{k} = [\ \mathbf{t} \vdots \mathbf{d}\ ] = [1, 0, 0, 0, ...,\ 1, \frac{1}{4}, ...]$$

In this way, the two parts of information are merged together, and the length of $\mathbf{k}$ is fixed, even though the number of distractors varies. A traditional REG task is, therefore, represented by a fixed-length vector, the REG processing can be performed by our NLG algorithm. The following steps sketch the process for doing this.

4.2 Extracting Schemas From a Corpus

So far, we represented REG tasks as knowledge vectors; this section extracts from the corpus a set of what we call expression schemas, or *schemas* for short. Both the knowledge vectors and schemas will be used to train the REG model. A schema represents the overall structure of an RE. For example, from an occurrence of "the red chair" in the corpus, we extract the schema

"the" [colour] [type]

Similarly, "the old man wearing glasses" derives the schema

"the" [age] "man" [hasGlasses]

The words are replaced according to the alignment labels in the training corpus. For example, in Tuna corpus, the words "red", "chair", "old", and "wearing glasses" are labelled with colour=*red*, type=*chair*, age=*old*, and hasGlasses=*true* respectively in the two sample expressions (Figure 2). In this way, we extract a schema from each corpus expression.

4.3 Schema Selection

This step uses schema selection to perform traditional REG attribute selection, among other

```xml
<description> the red chair </description>
<alignment>
  <word> the </word>
  <word attr="colour" value="red"> red </word>
  <word attr="type" value ="chair"> chair </word>
</alignment>
...
<description> the old man wearing glasses </description>
<alignment>
  <word> the </word>
  <word attr="age" value="old"> old </word>
  <word> man </word>
  <word attr="hasGlasses" value="true"> wearing glasses
  </word>
</alignment>
...
```

Figure 2: The (simplified) alignment examples for human-produced expressions in Tuna corpus in XML format.

things. It focusses on how to obtain a schema for a given REG task (represented by a knowledge vector i.e. our vectorised KR), focussing on what attributes should be expressed. Instead of creating new schemas, we reuse the existing schemas extracted from the training corpus. Because the schemas omit the specific attribute values, multiple expressions share the same schema (and multiple schemas share the same set of placeholders). To select a schema for a given RE task, we adopt the lexical selector of Li et al. (2016) as schema selector, which is fully statistical, and which accepts the knowledge vector.

To train the schema selector, we represent each unique schema i extracted from the training corpus as a column vector (denoted by $\mathbf{s}_i^T$), whose dimension is equal to the total number of data records in the training set $\mathcal{M}$. If the schema of the j-th data record of the corpus is same as schema i, then the j-th element of $\mathbf{s}_i^T$ equals 1, and 0 otherwise. In addition, we also need to construct a matrix (namely $\mathbf{K}$) which encodes the information of all the data records of the corpus. $\mathbf{K}$ consists of knowledge vectors of each feature group of the training data, with each row corresponding to a knowledge vector $\mathbf{k}$ of a data record. Suppose the training corpus includes m records and n features in total, then $\mathbf{K}$ is a $m - by - n$ matrix. Based on $\mathbf{K}$ and $\mathbf{s}_i^T$, our model training process finds a projection vector (i.e., the column vector $\mathbf{p}_i^T$) of Equation 3 by Least Square (Li et al., 2016).

$$\mathbf{K} \cdot \mathbf{p}_i^T = \mathbf{s}_i^T \tag{3}$$

$$\mathbf{p}_i^T = pinv(\mathbf{K}) \cdot \mathbf{s}_i^T \tag{4}$$

The projection vector $\mathbf{p}_i^T$ indicates how the information of a knowledge vector $\mathbf{k}_r$ projects on the use of the schema i. Therefore, a weight (w_r) that i should be adopted for expressing r is estimated by Equation 5.

$$w_r = \mathbf{k}_r \cdot \mathbf{p}_i^T \tag{5}$$

When every $\mathbf{p}_i^T$ is found, given an unseen knowledge vector ($\mathbf{k}^*$), we select the schema (denoted by x) for $\mathbf{k}^*$ such that x maximises $\mathbf{k}^* \cdot \mathbf{p}_{i=x}^T$ (Li et al., 2016):

$$x = \arg\max_x \left(\mathbf{k}^* \cdot \mathbf{p}_{i=x}^T\right) \tag{6}$$

When a schema is selected, we pick up and output the attributes of all the placeholders within the schema as the outcome for the REG task. Note that in the Tuna corpus, a given referent can never be described using two different values of the same attribute (e.g., a sofa cannot be both red and blue). Therefore, selecting attributes suffices for this part of the REG task.

In the above we have focussed on attribute selection. However, when a schema is selected, we do not only select attributes, we also fix the overall syntactic pattern of the RE. Thus, schema selection performs an important part of Linguistic Realisation as well.

4.4 Generating Complete Referential Noun Phrases

Expression schemas contain a lot of information about a referential Noun Phrase – including its syntactic structure and the use of function words – but they still contains placeholders for attributes. This section focuses on generating textual REs. The remaining generation task is to replace placeholders by actual words, after schema selection.

Analogous to schema selection, we adopt the selector of (i.e., Li et al., 2016) to do this. Placeholders are classified according to their corresponding attributes (e.g. `colour` or `type`). Two placeholders belong to the same class if they express the same attribute (whatever they are in the same or different REs), and we train an individual lexical selector for each class of placeholders.

For each placeholder class, we first build up a small training corpus that consists of all the words (or multi-word phrases) corresponding to the placeholder, with the features that the word (or the multi-word phrase) expresses.

Looking through the Tuna corpus, from the expressions, for example, "the red sofa" and "the blue chair", we find that the placeholder [colour] can be described by words "red" and "blue". So the extracted small corpus for [colour] contains the two words, which is shown in Table 3

Table 3: The extracted small corpus for placeholder [colour] from Tuna corpus

Text	Data
red	{`colour`=*`red`*}
blue	{`colour`=*`blue`*}
...	...

Then, for each class of placeholder (e.g. for [colour]), we train a lexical selector with the corresponding small corpus (e.g. Table 3). The training and word selecting processes are the same as in Schema Selection (§4.3). First, the words (and phrases) and corpus data in the small group are represented by vectors and a matrix; then they are used to find the projecting vectors through Equation 3; finally, words (or word phrases) are selected through Equation 6 for the placeholders of the class according to the given knowledge vector (that used to select the schema). Therefore, after selecting a schema, we replace its placeholders by the selected words (or word phrases) to transform the schema into a textual expression.

To summarise our approach to REG, we generate a textual RE (i.e., a Noun Phrase) for a REG task by adopting a three-step generation strategy: Given a REG task, we first represent the task as a knowledge vector which includes the knowledge of both target and distractor. Secondly, we select a schema, and finally, select words for each placeholder in the schema. If the process stops just after the schema selection, we still achieve the traditional RE task by picking up the attributes of the placeholder in the schema as the selected attribute set.

5 Evaluating the VB-REG Algorithm

We evaluated both Attribute Selection and Linguistic Realisation by making use of the Tuna corpus (Gatt et al., 2007), which contains two domains: Furniture and People. The Tuna corpus consists of corpus records; each record consists of a REG trail (i.e. one or two targets and some distractors) and a Noun Phrase produced by a participant to express the target. The Noun Phrases were collected in two conditions: the participants were either allowed to use location descriptions (e.g. "in the top left") or not (van der Sluis et al., 2006).

Here we focus on those corpus records which have one singular target object (rather than a set of two) and where the locational descriptions were not allowed to use. There are 210 corpus records in Furniture domain, and 180 records in People domain. We also discarded the records in which the locational descriptions were still used.

Our evaluation assumes a type of Knowledge Representation based on an attribute-value schema. Corpora such as ReferIt (Kazemzadeh et al., 2014), which do not use this type of KR, are not directly amenable to this approach.

5.1 Evaluating Attribute Selection in the VB-REG Algorithm

We evaluate the quality of our attribute selection using Dice score and PRP (Perfect Recall Percentage) scores, which are the most often used REG evaluation metrics (Van Deemter, 2016). Dice calculates the degree of similarity between sets: in our case, the set of properties expressed by the REs in the corpus versus the set of properties expressed by the REs generated by our algorithm; PRP gives the percentage of cases in which a generated RE expresses exactly the same properties as an RE in the corpus.

For each domain, we randomly divided the trials into two parts of the same size, for training and testing respectively. We repeated the experiment 10 times with different division of the trails to perform 10-fold cross-validation. We trained our model as explained in §4; then, we selected attributes for the testing data, and calculated the Dice and PRP scores based on the expressions of test data and the attributes generated. The scores of 10-fold cross-validation are shown in Table 4. The average Dice score (Mean) of the 10-fold experiments of Furniture domain is 0.916 with PRP being 61.6; for People domain, the average Dice score is 0.848 with PRP being 46.0.

To compare with other REG algorithms, we

Table 4: REG evaluation results, showing Dice scores, standard deviation (SD), PRP scores

	Furniture			People		
	Dice	SD	PRP	Dice	SD	PRP
fold 0	.930	.11	66.2	.828	.19	40.0
fold 1	.907	.13	60.3	.813	.22	44.8
fold 2	.917	.11	59.5	.854	.18	48.5
fold 3	.915	.12	60.6	.852	.18	46.6
fold 4	.914	.11	58.3	.885	.16	58.9
fold 5	.932	.12	69.2	.838	.17	44.8
fold 6	.924	.11	64.9	.832	.15	34.9
fold 7	.905	.13	60.3	.815	.17	36.4
fold 8	.923	.10	60.0	.892	.14	56.3
fold 9	.894	.14	56.6	.871	.15	49.2
Mean	.916	.12	61.6	.848	.17	46.0

Table 5: The performance comparison of our algorithm (VB-REG) with the Incremental Algorithms, Full Brevity algorithm, and Greedy Algorithm on Furniture domain.

	Dice	SD	PRP
IA-COS	.917	.12	6.9
IA-CSO	.917	.12	6.9
IA-RAND	.840	.15	31.4
IA-OCS	.829	.14	25.0
IA-SCO	.815	.14	19.2
IA-OSC	.803	.16	22.4
IA-SOC	.780	.16	18.6
FB	.841	.17	39.1
GR	.829	.17	37.2
VB-REG	.916	.12	61.6

show the tables from Van Deemter (2016, see Table 5 and Table 6) which shows the performance of the classic REG algorithms on the part of the Tuna corpus on which we focus (i.e., excluding location and references to sets): Full Brevity algorithm (FB), Greedy Algorithm (GR), and Incremental Algorithms (IA-xxx), the IA-xxx suffixes denote different Preference Orders. For example, IA-COS is the version for the furniture corpus that had colour (C), orientation (O), and size (S) as its first-most, second-most, and third-most preferred attribute; IA-GBHOATSS was the version of IA for the people corpus that used the Preference Order has_Glasses, has_Beards, Hair, etc.

Our approach, which is statistical and domain independent (hence does not distinguish between the furniture and people domain), preforms extremely well compared to the classic algorithms.

Table 6: Performance comparison of our algorithm (VB-REG) with the Incremental Algorithms, Full Brevity algorithm, and Greedy Algorithm on the People domain.

	Dice	SD	PRP
IA-GBHOATSS	.844	.17	44.7
IA-BGHOATSS	.822	.17	36.4
IA-GHBOATSS	.776	.21	29.5
IA-BHGOATSS	.728	.19	15.9
IA-HGBOATSS	.688	.18	3.8
IA-HBGOATSS	.658	.20	4.5
IA-RAND	.598	.23	11.4
IA-SSTAOHBG	.344	.11	0.0
FB	.764	.23	34.1
GR	.693	.20	8.3
VB-REG	.848	.17	46.0

Although our Mean Dice Score of Furniture domain (i.e. 0.916) is slightly lower than the champion, our algorithm beats the others in all other columns (Mean Dice scores of People domain and PRP for both domains).

5.2 Evaluating the fluency of NPs generated by the VB-REG algorithm

We also evaluate the fluency of the Noun Phrases generated by the VB-REG algorithm. We decided to use BLEU as a metric, with Noun Phrases produced by human experts as our baseline, using the same Tuna records and the same experimental settings as before. For each domain, the Tuna corpus was, once again, randomly divided into two parts: the training corpus and testing corpus.

The training corpus is only used to train the VB-REG model. After training, the model selects attributes and generates textual REs for each testing corpus data. The selected attributes are shown to 2 experts who are familiar with NLG but not with the Tuna corpus. The experts were also provided with relevant corpus texts. Then they were asked to produce referring expressions (i.e., Noun Phrases) for the attributes. They were asked to work individually, then discuss and produce only one answer sheet that both of them agreed on. The experts thus worked as a human Surface Realiser.

For each data in the testing corpus, we obtained three Noun Phrases: one generated by VB-REG algorithm, one produced by human experts; the third one was the original expression in the testing corpus. We calculated the BLEU scores of

Table 7: 10 fold REG evaluation results of BLEU scores

	Furniture		People	
	VB-REG	expert	VB-REG	expert
fold 0	.837	.889	.964	.923
fold 1	.751	.844	.910	.904
fold 2	.864	.815	.973	.923
fold 3	.873	.834	.931	.899
fold 4	.840	.844	.963	.888
fold 5	.822	.745	.979	.948
fold 6	.960	.793	.949	.977
fold 7	.853	.807	.883	.900
fold 8	.781	.901	.968	.897
fold 9	.840	.928	.983	.958
Mean	.842	.840	.950	.922
p-value	.943		.026	

the generated expression and expert expression by using the testing corpus (the corpus expression) as the reference for BLEU. Finally, we calculated the mean scores of every testing record. We performed 10-fold cross-validation; outcomes are shown in Table 7. The high BLEU scores suggest that our model achieves high fluency levels. In the furniture domain, our model performs at a similar level as the experts (no significant difference); in the people domain, our model "outperforms" our experts ($p = 0.026$). The p-values are calculated by a paired sample t-test.

6 Discussion

We have proposed a statistical model for automatically selecting attributes for REG and expressing these in an actual noun phrase. Our evaluation shows that the method performs well in terms of both attribute selection and text fluency.

Unlike previous approaches to REG (see e.g., §2), the VB-REG algorithm does not contain the idea of unique identification in any shape or form. Instead of singling out the referent from the distractors, our approach simply learns how human speakers refer. This approach has interesting consequences, not least for the dual phenomena of over- and under-specification, on which much work on REG has focussed. In a nutshell, our approach generates over- and under-specified REs to the extent that they occur in the data. Thus, if a corpus contains many underspecified REs (as may be the case if the corpus is based on children's speech, e.g., (Matthews et al., 2012)), then our REG algorithm will loyally reproduce these. If our corpus contains many highly over-specified REs (e.g. if the domain is complicated (Paraboni et al., 2007) or contains a lot of clutter (Koolen, 2013)), then so will our algorithm.

Referring Expressions Generation is more than the relatively simple reference task on which we have focussed here. For example, reference can use logical operators such as negation; reference can be to sets (including e.g. geographic regions (Turner et al., 2010)); it can involve gradable attributes; it can involve guesses about the hearer's knowledge (R.Kutlak et al., 2016); it can involve collaboration between speaker and hearer (Garoufi and Koller, 2014), and so on (Van Deemter, 2016). Although our evaluation has focussed on the singular part of the Tuna corpus only, we believe that the VB-REG approach is suitable for dealing with the above complications, provided some adaptations are made to our KR method (e.g. allowing us to represent how a *set* of target referents differs from all other domain elements). We hope to test this hypothesis in future.

VB-REG generates textual REs through schema selection and word selection. This generation strategy suffices for generating the type of REs found in the TUNA corpus, as we have seen, but syntactically complex REs can pose a problem. Consider the expression "the old man carrying a young dog", whose schema is

"the" [age] "man carrying a" [age] [animal]

, containing two placeholders for `age`. When VB-REG selects words for [age] (as in §3) above, it lacks the information that the [age] is for the man or the dog. In this case, VB-REG would select the same word for the two [age] placeholders. This limitation stems from the NLG approach in which VB-REG is embedded; if VB-REG is embedded into an NLG approach that does not have this limitation (e.g., providing a mechanism to distinguish the two [age] placeholders), the resulting algorithm will not suffer from this limitation.

In addition, VB-REG adopts the strategy that all the schema shares the same set of placeholders to adapt to small-scale corpus (e.g. Tuna), but the strategy may cause a syntactically error. Selection of lexicalisations for placeholders is done without taking the context of the schema into account. However, it is not guaranteed that all lexicalisation always fits into all placeholder (of different schemas). For example, we could have the following two textual descriptions:

"the man wearing glasses"
"the bespectacled man"

They would lead to the schemas:

"the" "man" [hasGlasses]
"the" [hasGlasses] "man"

where both "wearing glasses" and "bespectacled" present the candidate phrases for the placeholder [hasGlasses]. Thus the generated textual REs could be "the man bespectacled" or "the wearing glasses man", which are not syntactically correct. Although the strategy works well in our evaluation, it would be safe if a restrictive strategy is adopted. If the scale of training corpus allows, we can let each schema use an unique set of placeholder (i.e. placeholders are no longer shared by schemas). Because usually multiple corpus texts derive the same schema, the frequent schemas still obtain enough training data under the restrictive strategy.

7 Conclusion and Future Work

This paper has presented REG as a special case of NLG. It is important to note that the key strategy that we employed for representing an REG task – which separates properties of the referent from differences between the referent and the distractors – can be applied to any vector-based NLG approach, for example the neural-network-based NLG approaches (e.g. Wen et al., 2015) and Concept-based NLG approaches (e.g. Belz, 2008; Konstas and Lapata, 2013). In this way, these NLG approaches can perform REG by adopting our knowledge representation as their input. Neural-network-based approaches adopt fixed-length vectors as their input, and generate texts word by word, so our knowledge vectors and the corresponding corpus texts can be used to train them. Concept-text generation models can likewise adopt vectorised inputs. These approaches usually adopt concept sets (i.e. sets of pairs of attributes and values) as their input. Vector-based KR can be transformed into concept sets by regarding the vector entries as pairs of an entry index and an entry value. Thus our knowledge vectors can be adopted by these approaches.

One important item for future work is further evaluation. Although the model performs very well on the Tuna reference task, further experimental evidence on more challenging reference tasks is required in order to assess the generality of the proposed approach. It would be interesting, for instance, to apply the method to the ReferIt corpus (Kazemzadeh et al., 2014).

Acknowledgments

We gratefully acknowledge the anonymous reviewers for their very helpful comments.

References

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Robert Dale. 1989. Cooking up referring expressions. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 68–75. Association for Computational Linguistics.

Giuseppe Di Fabbrizio, Amanda J Stent, and Srinivas Bangalore. 2008. Referring expression generation using speaker-based attribute selection and trainable realization (attr). In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 211–214. Association for Computational Linguistics.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1914–1925, Seattle, Washington.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Konstantina Garoufi and Alexander Koller. 2014. Generation of effective referring expressions in situated context. *Language, Cognition and Neuroscience*, 29(8):986–1001.

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, Anja Belz, and Eric Kow. 2009. The tuna-reg challenge 2009: Overview and evaluation results. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 174–182. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

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*, pages 49–56. Association for Computational Linguistics.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Ruud Martinus Franciscus Koolen. 2013. Need i say more? on overspecification in definite reference.

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

Xiao Li. 2019. A fully statistical approach to natural language generation.

Xiao Li, Kees van Deemter, and Chenghua Lin. 2016. Statistics-based lexical choice for nlg from quantitative information. In *Proceedings of the 9th International Natural Language Generation conference*, pages 104–108.

Danielle Matthews, Jessica Butcher, Elena Lieven, and Michael Tomasello. 2012. Two-and four-year-olds learn to adapt referring expressions to context: effects of distracters and feedback on referential communication. *Topics in cognitive science*, 4(2):184–210.

Ivandré Paraboni, Kees Van Deemter, and Judith Masthoff. 2007. Generating referring expressions: Making referents easy to identify. *Computational linguistics*, 33(2):229–254.

Dhanesh Ramachandram and Graham W Taylor. 2017. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108.

Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 97–104. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

R.Kutlak, K. van Deemter, and C.Mellish. 2016. Production of referring expressions for an unknown audience: A computational model of communal common ground. *Frontiers in Psychology*, 7.

Ielka van der Sluis, Albert Gatt, and Kees van Deemter. 2006. Manual for tuna corpus: Referring expressions in two domains.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2002. Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data. *Computing Science Department, University of Aberdeen, Aberdeen, Scotland, Tech. Rep. AUCS/TR0201*.

Ross Turner, Somayajulu Sripada, and Ehud Reiter. 2010. Generating approximate geographic descriptions. In *Empirical methods in natural language generation*, pages 121–140. Springer.

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

Tsung-hsien Wen, Pei-hao Su, V David, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *In Proceedings of EMNLP. Association for Computational Linguistics*. Citeseer.