Article:

A Text Reassembling Approach to Natural Language Generation

Xiao Li, Kees van Deemter, and Chenghua Lin

Abstract

Recent years have seen a number of proposals for performing Natural Language Generation (NLG) based in large part on statistical techniques. Despite having many attractive features, we argue that these existing approaches nonetheless have some important drawbacks, sometimes because the approach in question is not fully statistical (i.e., relies on a certain amount of handcrafting), sometimes because the approach in question lacks transparency. Focussing on some of the key NLG tasks (namely Content Selection, Lexical Choice, and Linguistic Realisation), we propose a novel approach, called the Text Reassembling approach to NLG (TRG), which approaches the ideal of a purely statistical approach very closely, and which is at the same time highly transparent. We evaluate the TRG approach and discuss how TRG may be extended to deal with other NLG tasks, such as Document Structuring, and Aggregation. We discuss the strengths and limitations of TRG, concluding that the method may hold particular promise for domain experts who want to build an NLG system despite having little expertise in linguistics and NLG.

1 Introduction

Natural Language Generation (NLG) is the process of generating natural language texts from non-textual information. Until approximately the year 2000, NLG research focussed on generating texts from (broadly) logic-based meaning representations, but recent work has generated text from "flat" data, such as databases containing huge tables of sensor data, for instance in the weather domain. It is on this "data to text" type of NLG that this article will focus.

Different methods exist for performing (data to text) NLG. Among the earliest and most widely used are rule-based approaches (e.g. Reiter *et al.* (2009); Hunter *et al.* (2011); Mille *et al.* (2019)), which adopt a pipeline to break down the process of natural language generation into multiple subtasks that operate by using manually constructed rules. Another tradition of research focusses on the construction of syntactically structured templates for sentences and clauses, which are then filled and combined to produce texts, in accordance with a set of hand-coded rules (Van Deemter *et al.* , 2005; Braun *et al.* , 2019).

The behaviour of a rule-based NLG system is predictable, because it is governed by deterministic rules that are, to varying extents, readable by humans. When such a system generates textual descriptions for a set of input data, its behaviour and

outputs at each stage are fully tracable, as it is essentially a white box model. However, the shortcomings of rule-based approaches are also obvious and well documented. For example, the hand-crafting of rules costs an enormous amount of human effort.Furthermore, relying on human-crafted rules means that these approaches are almost always highly domain specific: it is difficult to generalise a rule-based NLG system to another domain, which might require a set of completely different rules. In the worst case, the design of the new system may have to be done almost entirely from scratch.

During the last 10-15 years, and especially in the last few years, a growing proportion of NLG research has concentrated on designing a new generation of *statistical* NLG approaches (Belz, 2008; Konstas & Lapata, 2013). These approaches learn knowledge from a data-text parallel corpus and this knowledge is then applied during the generation process. Although this new generation of NLG approaches are, at least in principle, highly generic, and although they can reduce manual NLG labour, they also have their limitations. First of all, as we shall argue below, the above-mentioned NLG systems do not yet completely abandon manually crafted rules.

For example, the system of Belz (2008) still relies on grammar rules. However, when training the system using both training text and a general grammar (such as English grammar), the grammar is often unable to cover every sentence structure in the training corpus. Consequently, the process of debugging the system involves a lot grammar rule revision. Although Konstas & Lapata (2013) did not use any grammar rules, their system still requires handcrafted structured descriptions for both the data and the generated texts. This greatly limits the ability of the system to be generalised: since the architecture of the system is based on a specific data structure, it is very difficult to migrate this system to other kind of data. More details of existing statistical NLG approaches will be discussed in Section 2.

Recent works on statistical NLG has explored the use of Deep-learning (e.g. Wen *et al.* , 2015; Li *et al.* , 2019; Dušek *et al.* , 2020). Deep learning-based approaches completely discard handcrafted rules and are capable of learning all the knowledge required for generating text from a single training corpus. However, deep neural networks are essentially black box systems; their overall logic is hidden behind large matrices. As such, these NLG approaches lose transparency. It happens frequently that a deep-learning based NLG system produces unexpected or incorrect output (Nie *et al.* , 2019; Dušek *et al.* , 2019), and the way in which these systems are set up means that it can be difficult to modify them in order to fix generation errors.

We believe that these issues have a considerable impact on the usefulness of NLG in the real world. An approach that does not have transparency (i.e. a black box approach) is difficult to change in light of requirements from clients. For example, a client may want the NLG program to always (or never) output certain sentences under certain circumstances. This is difficult to achieve by programs which have low transparency. It seems very likely to us that this is one of the reasons why statistical approaches have so far had only limited use in practically deployed NLG systems, especially in fault-critical applications.

Our aim with the work reported in this paper was to develop an approach to NLG

that does not require any handcrafted rules, and that is, at the same time, fully transparent. To achieve this aim, we developed the Text Reassembling Generation model (TRG). Unlike such previous approaches as Belz (2008) and Konstas & Lapata (2012), TRG learns to generate Textual Descriptions (TDs, which can be noun phrases, sentences, or multiple sentences) solely based on a standard concept-to-text corpus by reusing the text fragments in the corpus. Since the model does not depend on handcrafted rules, domain experts do not have to understand the rules of the language they are generating. Since the model is transparent, domain experts can trace how sentences are generated, and modify the model by hand. We argue that these features of TRG are theoretically desirable and potentially of great benefit to practitioners.

2 Related Work

While many current NLG systems are billed as statistical, we will argue here that many of these these are not completely abandoning the use of handcrafted rules. We also discuss NLG systems based on deep learning, which do abandon all handcrafted rules, but in a way that sacrifices transparency.

(Belz, 2008) introduced a framework called pCRU, which is an end-to-end statistical NLG approach. pCRU regards the entire NLG mission as an inversed semantic parsing process, called the expansion algorithm. Belz (2008)'s approach generates texts for the original data input based on a set of Context Free Grammars (CFGs). Probabilistic CFGs (pCFGs) are an extension of CFGs, where each CFG is assigned some probabilities indicating its frequency in the corpus. The pCRU framework then generates text following the probability distribution of CFGs. The probabilities, which are learned from a data-text corpus, drive a decision maker. During the generation process, inputs are expanded to grammar trees of the CFGs, and the decision maker governs the expansion process to estimate at which nodes the expansion should terminate. Strictly speaking, the pCRU framework is a semi-statistical NLG approach as it requires handcrafted CFGs for the base generator. A number of later approaches adopt a similar rationale to Belz's framework, based on a variety of grammatical formalisms, such as the Openccg framework based on Combinatory Categorial Grammar (White & Rajkumar, 2009, 2012), and Tree Adjoining Grammar (Gardent & Narayan, 2015).

Liang *et al.* (2009) introduced an automatic alignment approach, which aims to align the data representation to the texts within a data-text corpus. Although not designed for NLG originally, the approach of Liang *et al.* has impacted many NLG approaches as it can produce data and text alignments which are essential for NLG systems (e.g. Konstas & Lapata, 2012; Mei *et al.* , 2015; Gatt & Krahmer, 2018; Cao, 2020). For example, Angeli *et al.* (2010) proposed an approach which takes data to text alignment as input. The alignment information is used for extracting templates at multiple levels of granularities (i.e. sentence templates, expression templates, and word-level templates), and then texts are generated based on the templates in a hierarchical process.

While evidently very useful, Liang's work has some important limitations. First,

it's approach to alignment assumes that each data instance is expressed *at most once* in the text. Second, it assumes that each word in the text describes some data; this is problematic for function words (such as "the", "to"), since these link other words, instead of expressing any data by themselves. Consequently, function words have to be treated as part of other content words. In practice, Liang's approach does not provide a principled solution to decide which text segment (consisting of content words) the function words belong to, e.g., the text segment preceding or after the function words. Importantly, Liang's alignment method assumes that a word can only express one type of data or a single dimension of data (e.g., wind speed or wind direction): it can cannot handle situations in which words express multiple data dimensions, for instance as when the word "mild" expresses a combination of warm temperatures and low wind speed. These limitations limit the quality of data-to-text alignment.

Konstas & Lapata (2012, 2013) proposed a statistical end-to-end NLG approach, which extends the automatic alignment approach of Liang *et al.* (2009). Instead of simply using the alignment information, Konstas and Lapata recast the model as a simple probabilistic Context Free Grammar (pCFGs), which describes both the structures of the input data and the structures of texts. Then, the pCFGs are packed into a hypergraph; the weights in the hypergraph are learnt through an EM algorithm. In this way, the authors treat sentence generation as finding the maximally likely derivation tree. As Kanstas and Lapata builds upon Liang's work, it inherits the above-discussed limitations of that method. Their approach is also limited in dealing with data from the domains which are outside the coverage of the pCFG rules. While Kanstas & Lapata's approach is similar to ours in terms of overall architecture of alignment and generation, our proposed TRG model addresses their limitations without requiring any handcrafted rules.

Recent years have seen a surge of interest in applying Deep Neural Networks (DNN) to NLG related tasks such as dialogue generation (Wen *et al.*, 2015; Li *et al.*, 2017), referring expression generation (Liu *et al.*, 2017; Yu *et al.*, 2017), and language style transfer (Fu *et al.*, 2018; Yang *et al.*, 2018). The general learning paradigm of DNN based approaches for language is to learn some dense, low-dimensional vector representations, which capture the grammatical and semantic generalisations of the input text. Such representations, along with some tailored network architectures, are then applied to perform the task of interest.

Although DNN-based approaches have achieved remarkable success, they have significant drawbacks as well. For instance, deep learning approaches have been criticised for lacking in transparency (LeCun, 2017), i.e., difficult to interpret and functioning more or less as a black box. Consequently, deep learning approaches are difficult to relate to linguistic insights, and they are difficult to modify if and when this is needed. For instance, it's impossible to manually tweak or update a deep learning based decision support system say, when there is new regulatory changes that need to be taken into account: new training data which incorporates the new regulations would be required for retaining the model. Another well-known problem in deep learning approaches is *hallucination*, which is particularly prevalent in the language domain (Moosavi-Dezfooli *et al.*, 2017; Dusek *et al.*, 2019). In the
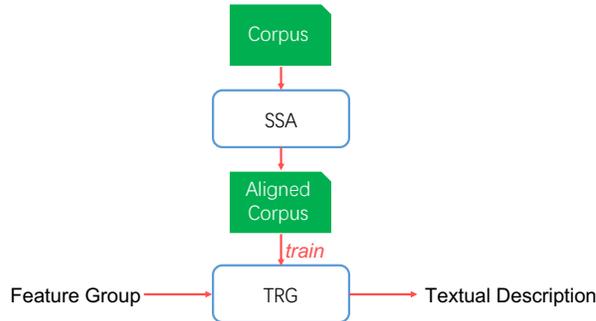
Fig. 1. The overall architecture of the TRG model. **The names in the Figure will be updated soon; SSA should be TRG-Alignment, TRG should be TRG-Generation**

context of NLG, hallucination means the generated text contains non-existent or incorrect information of input data. This is of course unacceptable in many real-world applications for instance when generating text that communicate important medical, financial, or engineering information. The recent E2E challenge (Dusek *et al.*, 2019) shows an example whose input data is "`name[Cotto]`, `eatType[coffee shop]`, `near[The Bakers]`", whereas the corresponding text generated by an NLG system was "*Cotto is a pub near The Bakers*", confusing a pub with a coffee shop. Another example is Lin *et al.* (2019), which proposed an end-to-end neural model for empathetic response generation. Despite the highly sensitive problem domain, the model sometimes generates potentially harmful responses (e.g., "*That is great!*") as a response to someone who talkes about a stressful situation he/she has been through. These problems are avoided by our approach.

3 Methodology

TRG is divided into two algorithms:

1. **TRG-Alignment** – an automatic alignment algorithm which produces a fine-grained alignment between text and data;
2. **TRG-Generation** – a fully statistical NLG algorithm.

Figure 1 shows the overall architecture of the TRG model. In a nutshell: **TRG-Generation** accepts data as input, and produces a text as output. Before generation, however, **TRG-Alignment** first has to ensure that the corpus contains a fine-grained alignment between text and data. Before explaining the working of these two algorithms in detail, section, let us first make explicit some of our background assumptions and define the main terms that will play a role in the remainder of this section.

3.1 TRG-Alignment

Let us state some assumptions (each of which was made by many authors before us), and define some key notions.

We assume the existence of what we call a *raw data-text corpus*. At the heart of this corpus lie, firstly, a sequence of Textual Descriptions and, secondly, a collection of concepts. A *Textual Description* (TD) is a coherent piece of text (for instance, a textual prediction of the weather for a given time interval and a given geographical location), made up of one or more sentences (e.g., "the temperature is mild"). A *concept* is the kind of information that a TD attempts to describe. We assume this information to be represented as a combination of a semantic attribute (e.g., temperature, or wind speed) and a value that this attribute can have (e.g., 20.5 degrees, or 5 knots).

Before TD-alignment, we assume each TD to be aligned with the concepts that it describes. Here is an example of a TD together with the *collection of concepts* (CC) with which the TD is aligned (see Figure 2 (a)).

Note that this alignment is still very course-grained. The task of the TRG-Alignment algorithm (algorithm 1 above) is to turn this course-grained alignment into a much more fine-grained type of alignment which we call *word-level alignment*. By this, we mean a many-many relation that links each concept in the corpus with that part of the TD that expresses it. More precisely, when a concept c is linked with a text fragment t, then t is the smallest contiguous part of the TD that describes c. Such a contiguous part of text we call a text fragment. More precisely, a *text fragment* is a sequence of words that appears consecutively in a TD. When TRG is done, the TD above is word-aligned with a collection of concepts; one example is Figure 2 (b):
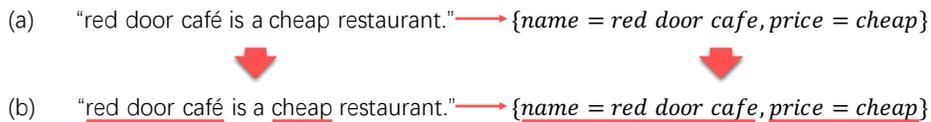
(a)   "red door café is a cheap restaurant." ⟶ {*name = red door cafe, price = cheap*}

(b)   "red door café is a cheap restaurant." ⟶ {*name = red door cafe, price = cheap*}

Fig. 2. The mapping between TD and feature collection of a raw concept-to-text corpus

In Figure 2 (b), the TD is split into four fragments, namely "red door café", "is a", "cheap", and "restaurant". The fragment "red door café" is aligned to the feature $name = red\ door\ cafe$, while "cheap" is mapped to $price = cheap$. Other fragments are not aligned to any feature (i.e., they are aligned to the empty set).

TRG-Alignment, as we have seen, relates fragments to concept. As we will explain below, this alignment is a many-to-many relationship, in which multiple fragments can be aligned with multiple concepts.

Unlike word-to-concept alignment in Liang *et al.* (2009), we can handle two common situations. Firstly, we can handle words that express multiple concepts, for instance as when the word 'muggy' expresses information about both temperature

and humidity (i.e.,m two different concepts). Secondly, we can handle situations in which non-contiguous text parts (i.e., two text fragments) express the same concept. For instance, in the TD "list flights from phoenix to san francisco, and arrive sfo before noon", both "san francisco" and "sfo" express the feature $to = san\ francisco$).

3.1.1 Fragment and Relationships between Fragments

Before we are getting into details, we define and clarify some key terms.

Text Fragment (or *fragment* for short). A fragment is a string of contiguous words within a TD, including any punctuation. For example, consider the TD that consist of the sentence "red door café is a cheap restaurant." This TD consists of 8 "words" (including the full stop), so the complete set of all text fragments within this TD has 36 elements (Figure 3).

red door café is a cheap restaurant .

1-gram: red, door, café, is, a, cheap, restaurant, .
2-gram: red door, door café, café is, is a, a cheap, cheap restaurant, restaurant .
3-gram: red door café, door café is, …, cheap restaurant .
   …   …
7-gram: red door café is a cheap restaurant, door café is a cheap restaurant .
8-gram: red door café is a cheap restaurant .

Fig. 3. *n*-grams derived from textual description "red door café is a cheap restaurant ."

If one fragment (b) contains another (a) as a part, we say that b includes a, denoted by $b \succ a$. Figure 4 shows the inclusion relationship between fragments of "red door café is a cheap restaurant.". This relationship is a lattice, representable as a triangle. The top vertex represents the longest fragment in the TD (i.e. the TD itself) and the bottom vertices are the words (i.e., unigrams). We shall call this the *fragment triangle*.

We define *neighbourhood of a fragment* according to the inclusion relationship. For a fragment w, we say fragment w' is the neighbourhood of w if w' includes w or w' includes w. For example, given "red door café is a cheap restaurant.", the neighbours of "café is a cheap" are:

<div align="center">

"door café is a cheap"

"café is a cheap restaurant"

"door café is a cheap restaurant"

...

"is a cheap"
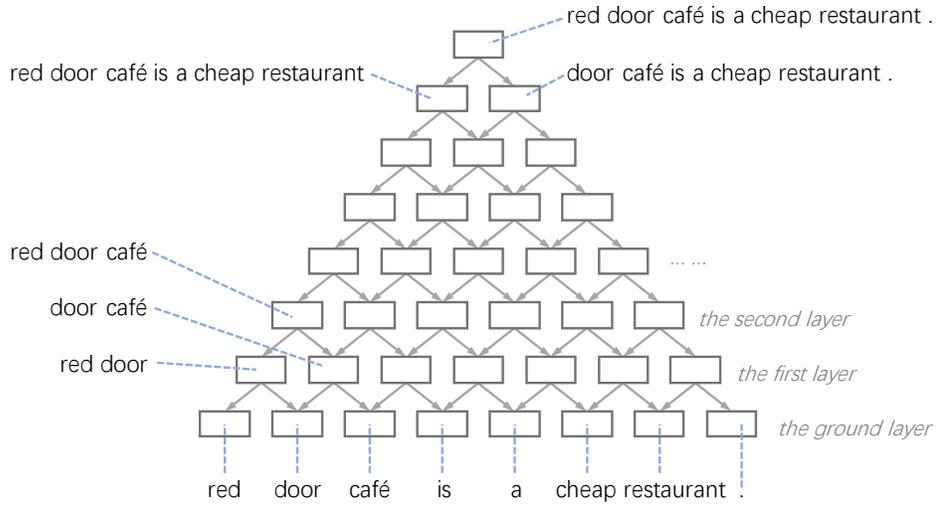
"café is a"

"is a"

...

</div>

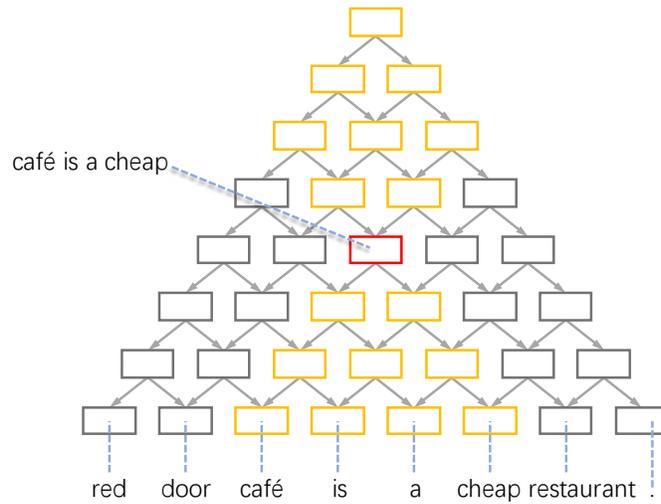Fig. 4. The fragment triangle of "red door café is a cheap restaurant."

Fig. 5. The neighbourhood (highlighted by amber border) of fragment "café is a cheap"

The amber boxes in Figure 5 are the complete neighbourhood of "café is a cheap".

3.1.2 Relationships between Fragments and Features

This section discusses the relation between a fragments (w) and features (g), which serves as the foundation for our alignment algorithm. We summarise the relationship in two questions:

1. Does fragment w expresses feature g?
2. To what degree does fragment w associate with feature g?

These two questions look similar at the first glance but they are different. If a fragment expresses a feature, the fragment represents the semantic meaning of the feature (e.g. "cheap" expresses $price = cheap$), but the fragment may also represent other semantic meanings which are not related to the feature. For example, the fragment "red door café is cheap" expresses the feature $price = cheap$, but it also expresses a different feature which is the café's name. We consider both relationships in our alignment algorithm. When comparing two fragments "red door café" and "red door café is cheap", "red door café" should be highly associated with feature $name = red\ door\ cafe$, but "red door café is cheap" is less so as it not only expresses the name of the restaurant but also the price. This section proposes two functions to present these two kind of relationships.

Modelling Relation, Part 1: Does fragment w express feature g?

The first issue that we face is how to model in terms of probabilities whether a fragment expresses a feature . If a fragment (w) expresses a feature (g), the conditional probability $P(g|w)$ should be close to 1. However, using $P(g|w)$ alone is not sufficient to accurately model whether a fragment expresses a feature. That is because if a feature frequently appears in the corpus (i.e. $P(g)$ is close to 1), for any fragment w, $P(g|w)$ will be close to 1.

To eliminate this effect, we define a function $Express(w, g)$ to model to what extent a fragment (w) expresses a feature (g). $Express(w, g)$ considers both $P(g)$ and $P(g|w)$; that is, it looks at the differences between $P(g|w)$ and $P(g)$ (i.e., ΔP) as well as how dominant the feature g is in the corpus (i.e., $1 - P(g)$):

$$
\begin{aligned}
Express(w, g) &= \frac{\Delta P}{1 - P(g)} \\
&= \frac{P(g|w) - P(g)}{1 - P(g)}
\end{aligned}
$$

We also show an intuitive illustration of the idea behind function $Express(w, g)$ in Figure 6.

If $P(g|w)$ is smaller than $P(g)$, it means that w does not express g. As probability cannot take negative values, we enforce that $Express(w, g) = 0$ when $P(g|w) < P(g)$ (Equation 1).

$$
Express(w, g) = \begin{cases} \dfrac{P(g|w) - P(g)}{1 - P(g)} & P(g|w) > P(g) \\ 0 & P(g|w) \leqslant P(g) \\ 0 & P(g) = 1 \end{cases} \tag{1}
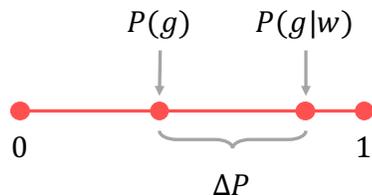$$

Fig. 6. Relationship among values of $P(g)$, $P(g|w)$, and ΔP

In addition, if a feature (g') appears in every corpus instance (i.e. $P(g') = 1$), the denominator becomes 0. Therefore we also define that $Express(w, g) = 0$ when $P(g) = 1$; in other words, if a feature appears in every instance of a corpus, the feature becomes unimportant or our model estimation.

$Express(w, g)$ can be used to estimate whether a fragment w is a *complete phrase* when w expresses a feature g. Here a *complete phrase* refers to a phrase which contains a full specification of feature g. Suppose a and b are two fragments expressing g, where a is a complete phrase (e.g., "red door café" or "red door café is"), whereas b is an incomplete phrase (e.g. "café is" or "door café"), the $Express(a, g)$ must be greater or equal to $Express(b, g)$. When a appears, b must appear because $a \succ b$; but when b appears, a may not appear, because b is an incomplete phrase which can also be part of other phrases used to express other features. For example, "door café" can not only constitute "red door café", but also "blue door café" or "white door café". As a result, the probability of $P(feature|incomplete\ phrase)$ is lower or equal to $P(feature|complete\ phrase)$. In other words, the complete phrase is more likely to express the feature than the incomplete phrase. For example, when we substitute "door café" and "red door café" into Equation 1, we have

$$P(g|\text{``door café''}) \leqslant P(g|\text{``red door café''})$$
$$\Rightarrow Express(g|\text{``door café''}) \leqslant Express(g|\text{``red door café''})$$

If a fragment (w) is not polysemous (w expresses the same features in any context), $Express(w, g) = 1$ where g is any feature that w expresses). That is, g always appears when w appears. Here we do not consider the case of polysemy, because we assume that a polysemous word can be disambiguated in context. That is to say, if a sentence contains a polysemous word, we only need to find and align fragments which contain both the polysemous word and the necessary context, and this fragment can still be regarded as non-polysemous.

Modelling Relation, Part 2: To what degree does fragment w associate with feature g?

The function $Express(w, g)$ above estimates the probability of a fragment (w) given a feature (g), but it cannot identify whether w contains extra words which express other information.

To align fragments to features, we identify and align the fragments that only express information about the corresponding feature. Fragments that express extra information (e.g., information about other features) are not used for alignment. For

example, both "red door café" and "red door café is" express $name = red\ door\ cafe$, but only "red door café" should be aligned to $name = red\ door\ cafe$. It should be noticed that we cannot always align the shortest fragment to the feature, because if a fragment expresses a feature, and the fragment contains a rare word, it is possible that the rare word can solely express the feature. Consider the restaurant named "the stinking rose" (which expresses $name = the\ stinking\ rose$). Here "stinking rose" contains enough information for $name = the\ stinking\ rose$, but we should still align "the stinking rose" to the feature, because in this case, "the" is a part of the name.

Therefore, we define the second function $Core(w, g)$ that adopts the probability $P(w|g)$ to estimate to what degree a fragment w associates with feature g (i.e. in what degree the use of w relies on g).

$$Core(w, g) = P(w|g) \tag{2}$$

Suppose each of the fragments w and w_+ expresses the feature g, and w does not contain any extra word which does not express g, while w_+ does, then we have:

$$Core(w, g) > Core(w_+, g) \tag{3}$$

Consider the fragments "cheap", "a cheap", and "cheap restaurant"; "cheap" expresses $price = cheap$, while "a" and "restaurant" do not. That is, the appearance of "a" and "restaurant" are events that are independent of the appearance of $price = cheap$. Therefore, we must have:

$$P(\text{"cheap"}|price = cheap) \geqslant P(\text{"a cheap"}|price = cheap)$$
$$P(\text{"cheap"}|price = cheap) \geqslant P(\text{"cheap restaurant"}|price = cheap)$$

In addition, if $Core(w, g) = Core(w_+, g)$, the degree to which the extra words in w_+ associate with g is equal to the degree with which w associates with g. The equality only holds if the extra words always appear when w appears. in which case the extra word should be considered as a part of the phrase that expresses g. For example, in "the stinking rose", the article is part of the name. For although "stinking rose" by itself can denote the restaurant (i.e. $Express(\text{"stinking rose"}, name = the\ stinking\ rose) = 1$), 'the stinking rose" is the fragment that aligns with $name = the\ stinking\ rose$, because

$$Core(\text{"stinking rose"}, name = the\ stinking\ rose)$$
$$= Core(\text{"the stinking rose"}, name = the\ stinking\ rose).$$

Whenever "stinking rose" appears, "the" must also appear.

In another example, "trattoria contadina" is a restaurant (in FS-Restaurant corpus), where both the words "trattoria" and "contadina" are only used to express the restaurant ($name = trattoria\ contadina$) in the corpus. In this case, not only

$$Express(\text{"trattoria contadina"}, name = trattoria\ contadina)$$
$$= Express(\text{"trattoria"}, name = trattoria\ contadina)$$
$$= Express(\text{"contadina"}, name = trattoria\ contadina)$$

but also

$$Core(\text{``trattoria contadina''}, name = trattoria\ contadina)$$
$$= Core(\text{``trattoria''}, name = trattoria\ contadina)$$
$$= Core(\text{``contadina''}, name = trattoria\ contadina)$$

The values of $Express(,)$ indicate that all the three fragments ("trattoria", "contadina" and "trattoria contadina") expresses $name = trattoria\ contadina$. However, the values of $Core(,)$ show all the words associate with $name = trattoria\ contadina$, so in the corresponding TD, "trattoria contadina" is the true fragment to express the feature, instead of "trattoria" or "contadina".

Overall, $Core(w, g)$ can test whether a fragment w is the shortest fragment that expresses a feature g, when w is known to express g. If w_+ contains words that do not express g, $Core(w|g) \geqslant Core(w_+|g)$; if $Core(w|g) = Core(w_+|g)$, we give up w and regard that w_+ truly expresses g.

3.1.3 Aligning Fragments to Features

Let us illustrate the alignment process by working through a concrete example. Each record (i.e. TD-CC pair)in the corpus is aligned, calling the functions $Express(w, g)$ and the $Core(w, g)$, each of which takes all data in the corpus into account.

What fragments align to what features depends on two types of information: whether the fragment expresses the feature, and to what extent the fragment associates with the feature.

$$weight(w, g) = Express(w, g) \cdot Core(w, g) \tag{4}$$

For each corpus instance (i.e. pair of TD and feature collection in training corpus), we firstly collect all fragments of the TD, and calculate the value of $weight(w, g)$ for each fragment (w) in the collection and each feature (g) in the feature collection.

Consider the FS-Restaurant corpus and focus on the instance in Table 1. The

Table 1. A real instance in FS-Restaurant corpus

TD	Feature Collection
"red door café is a cheap restaurant ."	$\{name = red\ door\ cafe, price = cheap\}$
...	...
...	...

TD expresses the feature collection with two features $name = red\ door\ cafe$ and $price = cheap$. So, we assume that fragments of the TD express a (possibly empty) set of features in the feature collectionWe separately substitute

$name = red\ door\ cafe$ and $price = cheap$ into $weight(w, g)$, and for each fragment (w) in the TD, we calculate the weights by Equation 4.

Figure 7, Figure 8, and Figure 9 show the outcomes of the functions $Express(w, g)$, $Core(w, g)$, and $weight(w, g)$ respectively, given that g presents $name = red\ door\ cafe$. The function values of each fragment (w) is marked in its vertex in each graph. The length of the background colour of a vertex also visualises the function value.
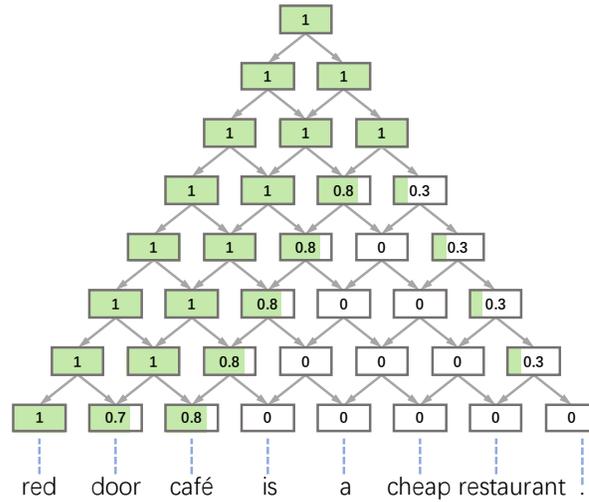

Fig. 7. Results of the values of $Express(w, g)$ for each fragment (w) in "red door café is a cheap restaurant ." when g is $name = red\ door\ cafe$

The outcomes of $Express(w, g)$ (in Figure 7) show that any fragment containing "red door café" has a large value (clode to 1), which is in line with our expectations, because any fragment containing "red door café" must express $name = red\ door\ cafe$. Because of the limited size of the corpus, the function values of "red", "red door", and "door café" are also large. So in this corpus, these three fragments are enough to distinguish (express) $name = red\ door\ cafe$. But they will not affect our aligning process; the following steps will eliminate this interference.

Comparing to $Express(w, g)$, outcomes of $Core(w, g)$ (in Figure 8) show that any fragment which "red door café" contains presents large results, because when the feature $name = red\ door\ cafe$ appears, these words are always used. At the same time, some other words including "is", "a", "restaurant", and the full stop present large outcomes as well, because they are the frequent words whether the feature appears or not.

We align the fragments to the feature according to the outcomes of the function $weight(,)$. If and only if both $Express(w, g)$ and $Core(w, g)$ return large outcomes, $weight(w, g)$ returns large outcome (see Figure 9). Based on the outcomes of $weight(w, g)$, we focus on the *maxima fragments* of the function. Given a feature g,
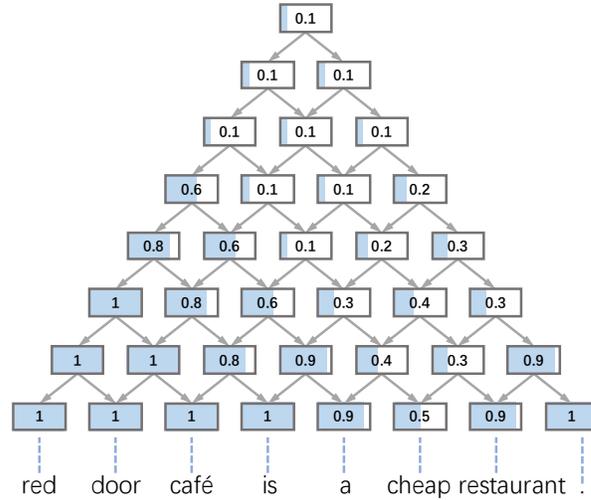
Fig. 8. Results of the values of $Core(w, g)$ for each fragment (w) in "red door café is a cheap restaurant ." when g is $name = red\ door\ cafe$
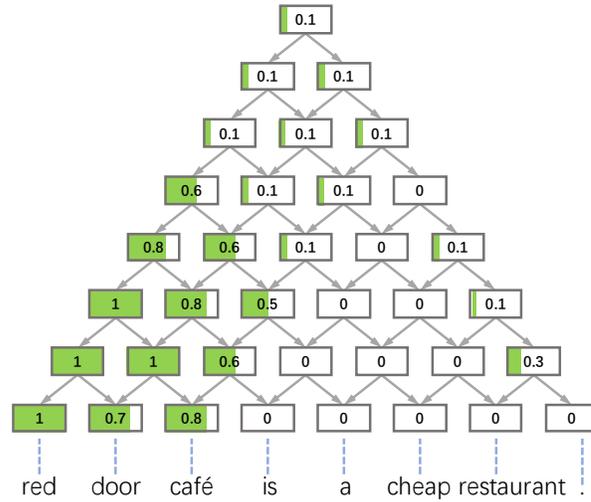
Fig. 9. Results of the values of $weight(w, g)$ for each fragment (w) in "red door café is a cheap restaurant ." when g is $name = red\ door\ cafe$

a maxima fragment is such a fragment (w) that for any neighbour fragment w' of w, $weight(w, g) \geqslant weight(w', g)$. Since we have marked the values of $weight(w, g)$ on the triangle (Figure 9), the maxima fragments can be intuitively explained through the triangle. Figure 10 highlights the maxima fragments by red border based on Figure 9. The maxima fragments may constitute a disconnected sub-graph, but vertices in each connected area of the sub-graph must share the same value (e.g. 1

Fig. 10. The highlighted maxima fragment region (in red box) for $weight(w, g)$, where g presents $name = red\ door\ cafe$. The red arrow points the fragment which is aligned to the feature.

and 0.3 in Figure 10). For each fragment in this area, we select and align the *longest* fragment to the feature. In addition, we discard the vertices whose value is lower than a pre-decided threshold $\sigma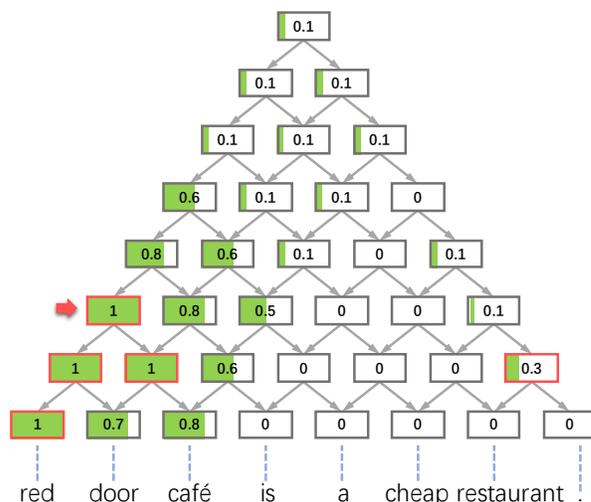$ (this thesis adopts $\sigma = 0.5$) to avoid the effects of noise in the corpus. Finally, in this case, only the fragment of "red door café" is aligned to $name = red\ door\ cafe$.

It is possible that multiple fragments align to one feature, that is, the feature is expressed multiple times in a TD by different phrases. Consider the following sentences in the Atis corpus:

"list flights from phoenix to san francisco, and arrive sfo before noon"

both "san francisco" and "sfo" express $to = san\ francisco$, so, both of them should be aligned to $to = san\ francisco$. Both of them present maxima values of function $weight(,)$ (see Figure 11).

The fragment triangle shown in Figure 11 still uses the same colour-based marking method. The green banners indicate the values of $weight(w, to = san\ francisco)$ for each fragment (w). The red borders indicate the maxima fragments, and the red arrows point the fragments which are the longest in each disconnected area, and which are selected by the method. It can be seen that this method accurately picked up all the fragments (viz. "san francisco" and "sfo") that express $to = san\ francisco$.

For each feature corresponding to a TD, we align the fragments to the feature respectively. When the fragments are aligned to all the features, the aligned fragments may be overlapped on each other. If two fragments overlap, we replace them with their union, and let the union fragment align to both features (see Figure 12).
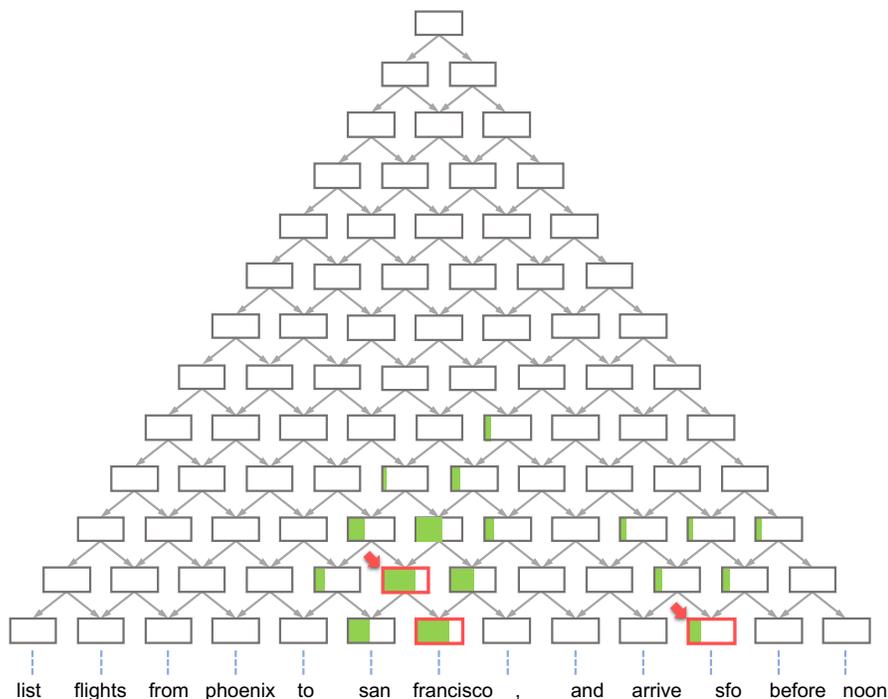
Fig. 11. The fragment triangle of "list flights from phoenix to san francisco, and arrive sfo before noon" in colour-based marking for the feature $to = san\ francisco$. The red arrows point the fragments which are aligned to the feature.

After the alignment, the TD is split according to the aligned fragments' borders. A
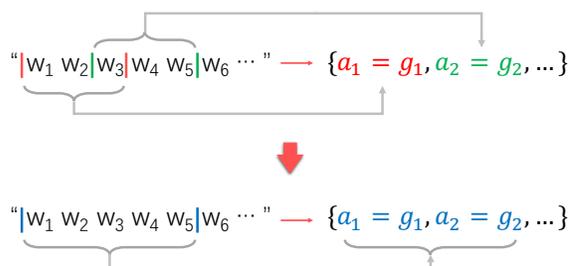
Fig. 12. Combining intersected segments

segment of the TD can be aligned to one or multiple features (e.g. "W_1 W_2 W_3 W_4 W_5" is aligned to $\{a_1 = g_1, a_2 = g_2\}$ in Figure 12), and can also be aligned to nothing (e.g. "W_6"). Therefore, the original alignment TD/feature-collection mapping is transformed to the segment/feature mapping. We apply the method for each corpus instance. After that, the corpus is transformed into the fragment-grained aligned corpus; each TD in the corpus is segmented, and each segment of a TD is

aligned to the features that it expresses (except the segments which do not express any feature).

3.2 TRG-Generation

In this section, we introduce the TRG-Generation component. It uses a generation strategy that we call "splitting-and-reassembling". Since the TRG-Alignment component has split the sentences into fragments, the generator only reuses and reassembles the fragments to construct the new sentences. The TRG-Generation component has it own training process: from the fine-grand aligned corpus, TRG-Generation component learns (1) what fragments express what features, and (2) how the fragments constitute a complete TD. For (1), since TRG-Alignment has already aligned the corpus, what fragments express what features is obvious to us. For (2), we regard all the sentences follows some *schemata*. We will introduce how the schemata are and how to extract them from sentences in Section 3.2.1.

The actual generation process is also a two step process, TRG-Generation (1) selects a schema as the sentence plan (Section 3.2.2), and (2) selects fragments to express given features and fills the schema with the fragments (Section 3.2.3). When we extract the schemata, we ensure when we fill the schemata with fragments, it always fulfil syntax correctness (for details seeing 3.2.1). When we select fragments, we ensure the fragments always express the given data. So, we generate the sentences with correct semantics and grammar.

3.2.1 Schema Extraction

Usually, a feature collection needs to be described by multiple fragments, that is, each fragment expresses a subset of feature in the collection. Therefore, in addition to learning what fragments express what features, the TRG-Generation component needs to learn at least two aspects of information. First, when a fragment of a feature collection is given, the model learns how to divide the feature collection into sub-collections so that each sub-collection can be expressed by a known fragment; second, the algorithm also needs to know how to reassemble the fragments into a grammatically correct TD. The TRG-Generation adopts a schema-based generation strategy to handle the problems.

Each TD in the corpus can be transformed into an *Expression Schema* (or saying schema for short). Recall that in the aligned corpus, the corpus TD have been split into segments according to the boundaries of fragments which are aligned to features. For each TD, we replace each segment of it with a placeholder, and each placeholder is associated with the concepts that the segment expresses. Here, the concepts are the attribute names of features, and the feature values are temporarily ignored. If a segment does not express any feature (e.g. they may be function words), we just leave them in the TD. For example, in the TD:

<p align="center">"red door café is a cheap restaurant."</p>

"red door café" and "cheap" are respectively aligned to $name = red\ door\ cafe$ and

price = cheap. We replace these two segments by [name] and [price], which denote
the two placeholders which correspond to *name* and *price* respectively. Other words
do not express any feature, so they are remained in the TD.

<p align="center">[name] "is a" [price] "restaurant ."</p>

After the replacement, the TD becomes the sequence of words and placeholders,
which is the expression schema in TRG-Generation component.

Since the segments are replaced by placeholders, multiple TDs derive the same
schema. Here we define that two schemas are the same if and only if all of their
sequence elements (words and placeholders) are correspondingly equal. For example,

<p align="center">"red door café is a cheap restaurant."
"trattoria contadina is a luxury restaurant."</p>

derive the same schema (i.e. [name] "is a" [price] "restaurant .").

Schemas present the two aspects of knowledge. First, they present how to decompose a given feature collection. Since each placeholder corresponds to a small group
of concept, an entire feature collection can be decomposed into such sub-collections
that each sub-collection consists of the features associated with the concepts of a
placeholder. Second, schemas also define how to joint the placeholders by words.
Thus, when we have the words (fragments) to fill the placeholders in a schema, the
schema can join them into a complete TD.

According to the extraction method, each TD in corpus can be converted into a
schema. Since multiple TDs correspond to the same schema, the total amount of
schema must be less than the number of TDs.

3.2.2 Schema Selection

Instead of creating new schemas, the TRG-Generation component reuses the existing schemas extracted from the training corpus. Because the schemas omit the
specific attribute values, multiple expressions share the same schema. Here, we
employ the selection method of Li *et al.* (2016).

To train a schema selector, we represent each unique schema i extracted from the
training corpus as a column vector of one-hot encoding (denoted by $\mathbf{s}_i^T$). In other
words, the dimension of $\mathbf{s}_i^T$ is equal to the total number of data instances in the
training set $\mathcal{M}$, and if the schema of the j-th data instance of the corpus is the same
as schema i, then the j-th element of $\mathbf{s}_i^T$ equals 1, and 0 otherwise. In addition, we
represent the relationship between schema and feature collection by constructing a
matrix (namely $\mathbf{K}$) according to all the data instances of the training corpus. $\mathbf{K}$
consists of knowledge vectors of each feature collection of the training data, with
each row corresponding to a knowledge vector $\mathbf{k}$ of a data instance. Suppose the
training corpus includes m instances and n features in total, then $\mathbf{K}$ is a $m - by - n$
matrix. Based on $\mathbf{K}$ and $\mathbf{s}_i^T$, we find a projection vector (i.e., the column vector
$\mathbf{p}_i^T$) of Equation 5. $\mathbf{p}_i^T$ can be found via Least Squares.

$$\mathbf{K} \cdot \mathbf{p}_i^T = \mathbf{s}_i^T \tag{5}$$

$$\mathbf{p}_i^T = pinv(\mathbf{K}) \cdot \mathbf{s}_i^T \tag{6}$$

When the $\mathbf{p}_i^T$ is found, selecting a schema for a given feature collection can be performed by the product of the vectorised feature collection and $\mathbf{p}_i^T$. We denote the schema selector by $sch(,)$. The projection vector $\mathbf{p}_i^T$ indicates how the information of a feature collection $\mathbf{k}_r$ projects on the use of the schema i. Therefore, a weight (w_r), which reflects whether schema i should be used to express r, is estimated by Equation 7.

$$w_r = sch(r, i) = \mathbf{k}_r \cdot \mathbf{p}_i^T \tag{7}$$

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

$$x = \arg\max_x (\mathbf{k}^* \cdot \mathbf{p}_{i=x}^T) \tag{8}$$

Equation 8 simply selects the most likely schema, which is so far so good. However, always selecting the most likely schema may not always produce the best TD. Thus, how to generate overall the best TD will be proposed later in this section.

It should be noted that although the placeholders in a schema only associate with concepts (i.e. feature attributes), the feature values are still considered when we select a schema for a given feature collection. Instead of just taking the concepts into account, in Equation 8, $\mathbf{k}^*$ and $\mathbf{p}_{i=x}^T$ are the feature vectors rather than concept vectors. This strategy of schema selection guarantees that if a particular schema should only be used to express particular features, it can be chosen if and only if the particular features appear. For example, in Sumtime-Meteo, when the wind speed is low, the wind forecasts ignore the expression of wind direction; hence, schema with no wind speed implies the low wind speed. Therefore, the schema selection should consider not only the concepts but also what features are contained in the feature collection.

3.2.3 Fragment Selection

Analogous to schema selection, we employ the same method again to perform the Fragment Selection task. For each placeholder of each schema, we train an individual fragment selector. Two placeholders are considered as same if and only if they are at the same location in the same schema, because what words (fragments) can fit in a particular placeholder may rely on the contexts of the placeholder. In other words, two placeholders for the same concept but in different schemas or at different locations in a same schema are regarded as different placeholders. For example, in the schema:

[name] "is a" [name] "."

the two placeholders of [name] are considered as different schemas. In the placeholders:

[name] "is a" [type] "."
[name] "are" [type] "."

the four placeholders of [name] and [type] are considered as four different placeholders.

Firstly, for each placeholder of each schema, we build up a small corpus that consists of all the fragments which fit the placeholder. The small training corpus of a placeholder is also a feature-text corpus, whose text data consists of all the fragment which derive the placeholder, and whose feature data is the features that the fragment express. For example, for the placeholder [name] in the schema "[name] is a [price] restaurant.", the fragments such as "red door café", "trattoria contadina" derive the placeholder. So the extracted small corpus for the placeholder contains the two fragments, which is shown in Table 2

Table 2. The extracted small corpus for placeholder [name] in the schema "[name] is a [price] restaurant."

Text	Data
red door café	$\{name = red\ door\ café\}$
trattoria contadina	$\{name = trattoria\ contadina\}$
...	...

Secondly, for each placeholder of each schema, we train a fragment selector with its corresponding small corpus (e.g. Table 2). The training process and fragment selection process are same as the processes of schema selection. The fragments and the features in the small corpus are represented by vectors and a matrix respectively; then they are used to find the projecting vectors through Equation 5; finally, fragments are selected through Equation 8 for the corresponding placeholder according to the given feature collection. In this way, after selecting a schema, we select fragments for each placeholder in the selected schema, and replace the placeholders by the selected fragments, so, the schema becomes a complete TD.

This generation strategy can largely guarantee the correct syntax of the generated TD. The reason is as follows: in general, the use of fragments which express specific features are independent to the schemas, such as the use of "red door café" is only related to whether $name = red\ door\ cafe$ appears. In contrast, if a fragment only fits a particular schema, it will be only selected when the schema is selected. The reason is as follows. Suppose a fragment only fits a particular schema; the schema should only appear when the fragment appears in the training corpus, that is, the co-occurrence probability of the fragment and the schema should be high. Since this fragment expresses particular features (otherwise it would be the words in a schema), the schema of the fragment also has the high co-occurrence probability with the features. Therefore, the TRG-Generation component will not break the consistency between the processes of fragment selection and schema selection. If the given feature collection results in the selection of the schema, the feature collection also results in the selection of the fragment.

3.2.4 Generation with Probabilities

Both schema selection and fragment selection involve probabilities, because their selectors return the distributions (of schema or fragment). In order to get the overall optimal TD, we need to consider the weights of the schema selection and the fragment selection of each placeholder. Recall that the schema selection and fragment selection always return the schema or fragment which has the highest weight (probability) according to Equation 8. Although the strategy works for selecting fragments, in the schema selection, we cannot always select the schema with the highest weight, because the schema may contain some placeholders which we cannot find a suitable fragment to fill.

When we select schemas, we should consider not only the weights of schemas, but also the weights of the fragments which would fit the placeholders inside the schema. That is, there are $n + 1$ weights for a generating TD whose schema has n placeholders. In the $n + 1$ weights, one is for the schema (denoted by $P(s)$), and the other weights (denoted by $P(s_1), ..., P(s_n)$) are for the fragments to fit the placeholders. For example, given the feature collection:

$$\{name = red\ door\ cafe, price = cheap\}$$

we may have the following weights:

$$
\begin{aligned}
&P(\text{[name] "is" [price] "restaurant ."}) &&= 0.9 \\
&P(\text{[name] "is" [type] "restaurant ."}) &&= 0.1 \\
&P(\text{[name]} = \text{"red door café"}) &&= 0.9 \\
&P(\text{[name]} = \text{"trattoria contadina"}) &&= 0.1 \\
&... &&...
\end{aligned}
$$

In order to select the overall best TD, we define the weight of a TD (denoted by $P(TD)$) as the minimum value of these $n + 1$ weights

$$P(TD) = Min(P(s), P(s_1), ..., P(s_n)) \tag{9}$$

From the perspective of sentence construction, the degree of a weight represents the confidence in selecting a fragment or a schema. In order to generate high quality TD, we should try to avoid any error in the TD. The minimum weight of these $n + 1$ weights is the most likely place to make errors. Therefore, this definition of the weight of TD reflects the probability that the most likely error occurs in the TD. Therefore, in order to select the TD with lowest error probability, we should select the schema (x) which maximises $P(TD)$.

$$x = \arg\max_x \left(Min(P(x), P(x_1), ..., P(x_n)) \right) \tag{10}$$

Overall, when we select a schema, what fragments will be selected should also be considered at the same time. The final selected schema should not only fulfil the requirement of expressing the given feature collection, but also ensure that we have appropriate fragments to fit its placeholders.

4 Evaluation

In this section, we evaluate how well the TRG Model equipped with our training algorithm can perform in terms of generating the syntactically correct and semantically correct sentences. We will compare the proposed model with two other recent approaches on three corpora.

For a quick review, the three concept-to-text corpora are: Atis (Konstas & Lapata, 2012), SF-Restaurant (Wen *et al.* , 2015), and Sumtime-Wind. Atis is a corpus containing flight information queries (with 4962 training and 448 testing sample points), which was from the Airline Travel Information Systems Dataset (Hemphill *et al.* , 1990), and transformed into the representation of concept-to-text corpus by Konstas & Lapata (2012). SF-Restaurant (Wen *et al.* , 2015) is a restaurant information querying corpus, wich has been splitted into training and testing datasets (4075 training and 1024 testing sample points).

Sumtime-Wind is derived from the Sumtime-Meteo (Sripada *et al.* , 2002) with 500 training and 500 testing sample points. Its Text Descriptions are the wind direction and wind speed segmentation, which are extracted from the original wind forecasts. For example, the original forecast

“W-SW LESS THAN 08 BACKING SSE 06-10 BY AFTERNOON”

is segmented into two new instances:

“W-SW LESS THAN 08”
“SSE 06-10”

The segments with their corresponding feature collections is manually extracted from Sumtime-Meteo corpus. The feature collections involve two quantitative concepts: wind direction and wind speed.

4.1 Experiment Design

In the experiment, we compared our approach with the approaches of Konstas & Lapata (2013) and Wen *et al.* (2015). Konstas' approach is a statistical approach driven by a simple probabilistic context-free grammar (pCFG). Wen's approach is a deep-learning-based approach, more precisely a Bi-directional LSTM.

We directly applied all of our approaches and Konstas' approach on the three corpora, and let them generate sentences for the testing data. The original plan was to apply all three approaches to all three corpora. However, it proved difficult to apply Wen's method to the Sumtime and ATIS corpora. Wen's approach is only applied on his own corpus (i.e. SF-Restaurant), because of specific to the dataset preprocessing steps. In Wen's approach, words which express features should be manually [1] replaced with the concept names from the corpus texts. For example:

---

[1] Wen *et al.* (2015) said the replacement is automatic rather than manual. However, Wen's approach assumes that the concept values must reappear in the TDs. For example, if a feature collection includes *name = zen yai thai restaurant*, then the Text Description has to follow the form: “... zen yai thai restaurant ...”. Thus, the replace-

"red door café is a cheap restaurant."

should be changed to

"SLOT_NAME is a SLOT_PRICE restaurant."

Therefore, Wen's approach is not easy to migrate to other corpora. The TRG model and Konstas' approach were applied on the original SF-Restaurant corpus without the replacement.

We downloaded Wen's code from his website [2] and followed the procedures specified there; other than the issue mentioned above, we did not meet any problems. We also downloaded Konstas's code from his website[3]; since this web site did not contain explicit procedures for training and testing, we asked the authors for clarification, then strictly followed the procedures they kindly provided us with.

For all three approaches, we first evaluated the syntactic correctness of the sentences generated through grammar checkers. The generated sentences (with the human-written sentences) were judged by the grammar checkers and we counted how many sentences pass without syntax errors. Two general grammar checkers were applied: Language Tool[4] (LT) – a commercial grammar checker; and Language Checker[5] (LC) – an open-source grammar checker. The sentences in Atis and SF-Restaurant are general English so we judged them by LT and LC. However, sentences in Sumtime-Wind are a somewhat formulaeic sub-language that differs from everyday English, so to judge these we built a simple special-purpose grammar checker (denoted as WG), focusing on wind forecasts.

Secondly, we also evaluated syntactic correctness by human judges. Participants were asked to rate the generated sentences on a 10-point scale, rating how well they thought the generated sentences were syntactically correct (with 0 being completely incorrect and 9 being the most accurate).

Finally, we designed another human experiment to evaluate semantic correctness. We paired each generated sentence with a human-written sentence (as the golden standard); the human-written sentence pairs the input data of the generated sentence in the corpus. We provided the participants with the generated sentences and human-written sentences together, and the participants were indicated which is the human-written sentence. Then, for each sentence pair, we asked the participants to rate *to what extent the generated sentences express the same information as the human-written sentences* on an 11-point rating scale (0 means the two sentences express something entirely different; 10 means they express the exact same information).

The human experiments (both the one about syntactic correctness and the one

---

ment just finds and replaces words which are the reappearance of the feature value in the texts. For the feature values which do not reappear in the text, Wen's approach manually coded a detailed mapping e.g. "allow kid -s" maps to SLOT_KIDALLOW etc.

[2] `https://github.com/shawnwun/RNNLG`

[3] `http://www.ikonstas.net/index.php?page=resources`

[4] `https://languagetool.org/`

[5] `https://pypi.python.org/pypi/language-check`

about semantic correctness) involved 7 participants, postgraduate students in Computer Science (4 participants) and in Architecture (3 participants). In each of the two experiments, participants were asked to rate as many sentences as possible (using randomly sampled sentences). Each participant in the syntax experiment rated 10-30 sentences from each of the relevant generators in each of the relevant domains (SF-Restaurant, Atis, and Sumtime), and the same is true for the semantics experiment. In total around a thousand examples were rated in each experiment (1364 syntax experiment; 986 semantics experiment).
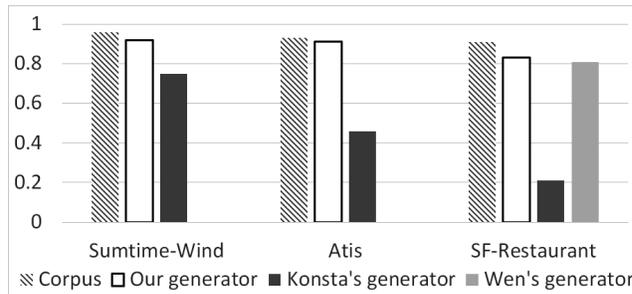

Fig. 13. Human evaluation on syntactic correctness.
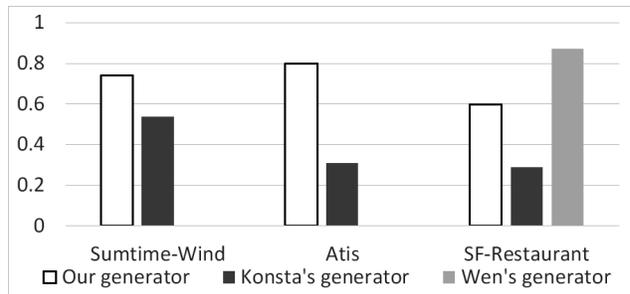
Fig. 14. Human evaluation on semantic correctness.

4.2 Results

Syntactic evaluation results are presented in Table 3, and Figure 13, while semantic evaluation results are in Table 4, which is visualized in Figure 14. All the scores from both grammar checkers and human raters were normalized to the decimals between 0 - 1.

The automatic and human results for syntactic evaluation suggest that our generated sentences perform equally well as the human-written sentences. While the overall performance gain of our proposed approach over the approach of Konstas is significant ($p < 0.02$ by two-tailed t-test), our approach and Wen's were statistically indistinguishable. In the semantic evaluation task, our approach outperformed

Table 3. Syntactic evaluation results (* indicates the performance significantly outcomes the baselines (excluding human-written) in the table with $p < 0.01$ by a paired sample t-test).

	Sumtime-Wind		Atis			SF-Restaurant		
	WG	Human	LT	LC	Human	LT	LC	Human
Human-written	0.99	0.96	0.97	0.95	0.93	0.95	0.96	0.90
Our generator	**0.98***	**0.92***	**0.96**	0.94	**0.91***	**0.95***	**0.96***	**0.83**
Konstas' generator	0.54	0.75	0.92	**0.96**	0.46	0.65	0.62	0.21
Wen's generator	–	–	–	–	–	0.88	0.88	0.81

Table 4. Semantic evaluation results (* indicates the performance significantly outcomes the baselines in the table with $p < 0.001$ by a paired sample t-test).

	Sumtime-Wind	Atis	SF-Restaurant
Our generator	**0.74***	**0.80***	0.60
Konstas' generator	0.54	0.31	0.29
Wen's generator	–	–	**0.88**

Konstas' approach ($p < 0.03$ by Mann-Whitney U-test for the Likert scales). Wen's model appears to outperform ours, but the results were not statistically significant. It also should be noticed that Wen's approach is trained on the corpus with replacement, which is much easier than the original corpus.

Both the approaches of us and Konstas performed well on the Sumtime-Wind compared with their performances on SF-Restaurant respectively. It is probably because the sentences in Sumtime-Wind are simpler than in SF-Restaurant with just two classes of concepts.

In the experiments, Konstas' approach gains surprisingly low scores. We believe this might be for the following reason. In Konstas' experiments, both a gold standard (the human-written sentences) and a basic baseline (sentences generated by a unigram model) are involved, and they were provided to the human participants together with the sentences generated by Konstas' approach. Because sentences of the unigram baseline are almost unreadable; they define a very low baseline, to distinguish with it, participants would produce the higher scores to Konstas' approach. However, in our experiment, we did not involve a basic baseline, but we did involve the gold standard (the human-written sentences). Consequently, our participants tended to rate them by the stricter standard.

5 Discussion

In this paper, we have presented a fully statistical approach to NLG which is based on a extraction of placeholders and schemas. Evaluation results on three different corpora suggest that our approach performs well, both in terms of syntactic correctness (based on both automatic metrics and human judgements) and in terms of semantic correctness (based on human judgments). The method generally outperformed that of Konstas & Lapata (2013), even on the ATIS corpus, for which their method was originally developed. Comparing the performance of our method against Wen *et al.* (2015) is proved to be more difficult, because Wen's method could only be applied to one of the three corpora that we studied; on this one corpus (i.e., the one developed by the authors themselves), however, Wen's method outperformed ours.

Researchers have employed statistically inspired approaches to NLG for a number of years now, but we have argued that many of the proposed approaches have not been *fully* statistical. The approach of Belz (2008), for example, relied on a pCFG that was trained from a hand-written CFG. In some cases, as we have seen in the section of experiment design, even the approach of Wen *et al.* (2015), which aims to be purely statistical, appears to rely on alignment procedures that are not fully specified. The approach proposed in the present paper does away with all handcoding and offers a fully statistical approach to NLG.

We believe, moreover, that it is an important advantage of our approach that it is *inspectable* to a much greater extent than many of its competitors. In particular, it is possible to inspect what words (fragments) correspond to what placeholders, and to change them to modify the behaviour of the system if this is desired; similarly, it is possible to scrutinise the set of schemas, and to modify one or more schemas in light of new evidence. After training, what TRG model learns is the generation rules – the schemas and fragments, which could still be easily modified by human users. Through reading the schemas and fragments, human users can generally understand what sentences the rules result. If particular sentences should be generated or be avoid to generate, manually adding or removing the relative schemas and fragments is enough to do so. A lack of inspectability is often cited as one of the main drawbacks of approaches based on Neural Networks LeCun (2017). After training, what the neural-network-based approaches learn is massive weights of matrices. Human users cannot trace what particular sentences can be generated according to the matrices. Consequently, to modify a trained approach is impossible.

A limitation of the work presented here is that the training algorithm we proposed does not use the training data efficiently. The TRG algorithm is very harsh on the assumption of placeholder equality. Only the same placeholders of the same schema can share a small corpus. This strict criterion is to ensure the syntactic correctness of the generated TD, because words expressing the same feature may have different syntactic roles (i.e. word forms, e.g. none, adj, adv, etc.). However, this strict criterion also makes the division of small corpora being too fine. When the

training corpus is small, the placeholders in the unpopular schema cannot obtain the acceptable size of the small corpus.

Although the possibility of lacking data potentially limits the diversity of TD generated by our algorithm under small dataset, the three corpora used in our evaluation are large enough, so our algorithm still performs. The TRG model can still learn enough schemas and fragments to perform the task of the TD generation. Meanwhile, when the selectors selects schemas and fragments, it gives priority to the schemas and fragments with more occurrences. So the limitation does not (or rarely) disturb our evaluation.

6 Conclusion

This paper proposed a purely statistical NLG approach – TRG model. Given a concept-to-text corpus, the approach can automatically analyse and learn how TDs in the corpus express the corresponding data (feature collections), and generates new TDs by imitating the corpus TDs. The algorithm generates TDs based on the split-reassemble strategy. We first decompose the TDs in the corpus into fragments and schemas by calculating the fragment-feature alignment. In the aligning process, we calculate what fragments express what features, and align the most appropriate fragments to the corresponding features. When the model generates a TD for a new feature collection, the model selects the schema and fills the placeholders of the schema with fragments to get the complete TD. In the evaluation, we verified that the generated TD has good ideographic ability and syntactic correctness based on three different data-text corpora. In the future, we will continue to improve the model and the efficiency of the schema extraction algorithm, such as trying to detect and merge small corpora so that each placeholder gets more sufficient candidate fragments.

This approach combines the advantages of machine-learning and rule-based NLG approaches. It is considered as the combination of a statistical method for rule extraction with a rule-based generation process, and the learnt schemas and the semantically meaningful fragments can be considered as the generation rules. This model presents the automatic training process and also keeps the transparency. The advantages of automation allow our program to learn the given corpus without any process of manually analysing and summarising the rules or templates for generation like what rule-based NLG programs have to do. Transparency makes the learnt results of the approach (i.e. schemas and the semantically meaningful fragments) easily modified by hand, so the performance of TRG model can be easily predicted and corrected. This ability is especially important for NLG programs to become commercial software, and this is exactly what the machine-learning NLG approaches do not keep.

References

Angeli, Gabor, Liang, Percy, & Klein, Dan. 2010. A simple domain-independent probabilistic approach to generation. *Pages 502–512 of: Proceedings of the 2010*

Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

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

Braun, Daniel, Klimt, Kira, Schneider, Daniela, & Matthes, Florian. 2019. SimpleNLG-DE: Adapting SimpleNLG 4 to German. *Pages 415–420 of: Proceedings of the 12th International Conference on Natural Language Generation.*

Cao, Juan. 2020. Generating Natural Language Descriptions From Tables. *IEEE Access*, **8**, 46206–46216.

Dusek, Ondrej, Novikova, Jekaterina, & Rieser, Verena. 2019. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.07931.*

Dušek, Ondřej, Howcroft, David M, & Rieser, Verena. 2019. Semantic Noise Matters for Neural Natural Language Generation. *arXiv preprint arXiv:1911.03905.*

Dušek, Ondřej, Novikova, Jekaterina, & Rieser, Verena. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *Computer Speech & Language*, **59**, 123–156.

Fu, Zhenxin, Tan, Xiaoye, Peng, Nanyun, Zhao, Dongyan, & Yan, Rui. 2018. Style transfer in text: Exploration and evaluation. *In: Thirty-Second AAAI Conference on Artificial Intelligence.*

Gardent, Claire, & Narayan, Shashi. 2015. Multiple adjunction in feature-based tree-adjoining grammar. *Computational Linguistics*, **41**(1), 41–70.

Gatt, Albert, & Krahmer, Emiel. 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.

Hemphill, Charles T, Godfrey, John J, Doddington, George R, *et al.* . 1990. The ATIS spoken language systems pilot corpus. *Pages 96–101 of: Proceedings of the DARPA speech and natural language workshop.*

Hunter, James, Freer, Yvonne, Gatt, Albert, Reiter, Ehud, Sripada, Somayajulu, Sykes, Cindy, & Westwater, Dave. 2011. BT-Nurse: computer generation of natural language shift summaries from complex heterogeneous medical data. *Journal of the American Medical Informatics Association*, **18**(5), 621–624.

Konstas, Ioannis, & Lapata, Mirella. 2012. Unsupervised concept-to-text generation with hypergraphs. *Pages 752–761 of: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics.

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

LeCun, Yann. 2017. *What's wrong with deep learning?* `http://techtalks.tv/talks/whats-wrong-with-deep-learning/61639/`. Accessed September 11, 2017.

Li, Jiwei, Monroe, Will, Shi, Tianlin, Jean, Sébastien, Ritter, Alan, & Jurafsky, Dan. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547.*

Li, Ruizhe, Li, Xiao, Lin, Chenghua, Collinson, Matthew, & Mao, Rui. 2019. A Stable Variational Autoencoder for Text Modelling. *In: The 12th International Conference on Natural Language Generation (INLG 2019)*. ACL Anthology.

Li, Xiao, van Deemter, Kees, & Lin, Chenghua. 2016. Statistics-Based Lexical Choice for NLG from Quantitative Information. *Pages 104–108 of: Proceedings of the 9th International Natural Language Generation conference.*

Liang, Percy, Jordan, Michael I, & Klein, Dan. 2009. Learning semantic correspondences with less supervision. *Pages 91–99 of: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1.* Association for Computational Linguistics.

Lin, Zhaojiang, Madotto, Andrea, Shin, Jamin, Xu, Peng, & Fung, Pascale. 2019. MoEL: Mixture of Empathetic Listeners. *arXiv preprint arXiv:1908.07687.*

Liu, Jingyu, Wang, Liang, & Yang, Ming-Hsuan. 2017. Referring expression generation and comprehension via attributes. *Pages 4856–4864 of: Proceedings of the IEEE International Conference on Computer Vision.*

Mei, Hongyuan, Bansal, Mohit, & Walter, Matthew R. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838.*

Mille, Simon, Dasiopoulou, Stamatia, & Wanner, Leo. 2019. A portable grammar-based NLG system for verbalization of structured data. *Pages 1054–1056 of: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing.*

Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein, Fawzi, Omar, & Frossard, Pascal. 2017. Universal adversarial perturbations. *Pages 1765–1773 of: Proceedings of the IEEE conference on computer vision and pattern recognition.*

Nie, Feng, Yao, Jin-Ge, Wang, Jinpeng, Pan, Rong, & Lin, Chin-Yew. 2019. A simple recipe towards reducing hallucination in neural surface realisation. *Pages 2673–2679 of: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.*

Reiter, Ehud, Turner, Ross, Alm, Norman, Black, Rolf, Dempster, Martin, & Waller, Annalu. 2009. Using NLG to help language-impaired users tell stories and participate in social dialogues. *Pages 1–8 of: Proceedings of the 12th European Workshop on Natural Language Generation.* Association for Computational Linguistics.

Sripada, Somayajulu, Reiter, Ehud, Hunter, Jim, & Yu, Jin. 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.*

Van Deemter, Kees, Theune, Mariët, & Krahmer, Emiel. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics,* **31**(1), 15–24.

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

White, Michael, & Rajkumar, Rajakrishnan. 2009. Perceptron reranking for CCG

realization. *Pages 410–419 of: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1.* Association for Computational Linguistics.

White, Michael, & Rajkumar, Rajakrishnan. 2012. Minimal dependency length in realization ranking. *Pages 244–255 of: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning.* Association for Computational Linguistics.

Yang, Zichao, Hu, Zhiting, Dyer, Chris, Xing, Eric P, & Berg-Kirkpatrick, Taylor. 2018. Unsupervised text style transfer using language models as discriminators. *Pages 7287–7298 of: Advances in Neural Information Processing Systems.*

Yu, Licheng, Tan, Hao, Bansal, Mohit, & Berg, Tamara L. 2017. A joint speaker-listener-reinforcer model for referring expressions. *Pages 7282–7290 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*