# FORGe at E2E 2017

**Simon Mille and Stamatia Dasiopoulou**
Universitat Pompeu Fabra, Roc Boronat 138, 08018 Barcelona, Spain
firstname.lastname@upf.edu

## Abstract

This paper describes the FORGe generator at E2E. The input triples are mapped onto sentences by applying a series of rule-based graph-transducers and aggregation grammars to template predicate-argument structures associated to each property. We submitted two primary systems to the task, one based on the grammars, and one based on templates, and one secondary system, which is a variation of the grammar-based one.

## 1 Introduction

The FORGe rule-based generator was originally designed to take linguistic predicate-argument structures as input. It has been further developed in order to support RDF triples as input in the framework of the WebNLG challenge (Gardent et al., 2017), in which the task consisted in generating texts from up to 7 DBpedia triples from 9 categories[1] and covering 373 different DBpedia properties.

The input for the E2E shared task (Novikova et al., 2017) is very similar to the WebNLG one in the sense that it consists of a list of up to 8 triples [2] corresponding to 8 different properties from the Restaurant domain. However, the task is slightly different as, while the reference texts are in most cases well aligned with the input properties, they also can contain (i) not all the information present in the input, and (ii) some information that is not present in the input. Consider for example the properties/values list found in the training data:

name[Strada]
*eatType*[pub]
*food*[English]
*priceRange*[moderate]
*customerRating*[average]
*near*[Yippee Noodle Bar]

And one corresponding expected generated text: *Come with your best friend to Strada restaurant and eat the best food in town near the Yippee Noodle Bar*. On the one hand, the properties *food[English]*, *eatType[pub]* and *customerRating[average]* are not verbalized in the generated text; on the other hand, the fact that Strada serves the best food in town, or that you should go there with a friend is not explicitly encoded in the input structure.

From this perspective, the E2E challenge is not a typical NLG task. It focuses on the (many) different ways to render a set of triples starting from exactly the same information, and thus addresses primarily data-driven systems. The E2E data consists of about 50K instances; 108 different combinations of properties are found in the training set, which gives an average of about 500 references by set of properties. The evaluation data does not contain new combinations of properties, only of property values.

The FORGe generator is a rule-based system that generates all and only the input properties.[3]. We submitted 3 outputs for evaluation:

- E2E_UPF_1 is the output of FORGe with the complete set of aggregation rules (Primary system 1);
- E2E_UPF_2 is the output FORGe with a selected subset of aggregation rules;
- E2E_UPF_3 is a reference output that consists of handcrafted templates filled with the val-

[1] Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam and WrittenWork.

[2] All properties can be seen as triples, with their value as object and the restaurant itself as the subject.

[3] Note that it would be possible to couple it with a statistical module that indicates which content to trim or add; this has not been done in the context of this challenge.

ues found for each property (Primary system 2).

Sections 2 to 5 describe the main system; the generation process involves the following steps: (i) mapping of properties onto Predicate-Argument (PredArg) templates (Section 2); (ii) template population (Section 3); (iii) sentence planning (Section 4); (iv) linguistic generation (Section 5). Section 6 briefly describes the approach for the full template-based approach followed for E2E_UPF_3, and Section 8 discusses the results of the evaluation.

## 2 Mapping of properties to PredArg templates

As stated in the Introduction, we consider the properties of the dataset and their values as triples, taking as subject the value of the *name[]* property, and as object their own value. We defined 32 handcrafted predicate-argument templates; in order to allow for some variation in the way the properties are verbalized, we associated (also manually) each of the 7 properties (8 minus *name[]*) found in the training data to one or more of these templates:

| Property | # |
|---|---|
| area[] | 4 |
| customerRating[] | 3 |
| eatType[] | 1 |
| familyFriendly[] | 11 |
| food[] | 4 |
| near[] | 4 |
| priceRange[] | 5 |

Table 1: Number of predicate argument template for each property

Parts of speech (e.g., NP (proper noun)), grammatical features (e.g., verbal tense or nominal definiteness), or, e.g., classes, for instance, can be specified in the template.[4] Figure 1 shows sample PredArg templates associated to the properties *near* and *customerRating* respectively.

## 3 Population of the templates

Using the aforementioned mappings between properties and predicative-argument templates, each input entry is transformed into a respective set of PredArg structures. First, for each considered triple, we find the set of admissible templates by looking at the combination of the respective

[4]Unspecified values are assigned as needed later in the generation process.

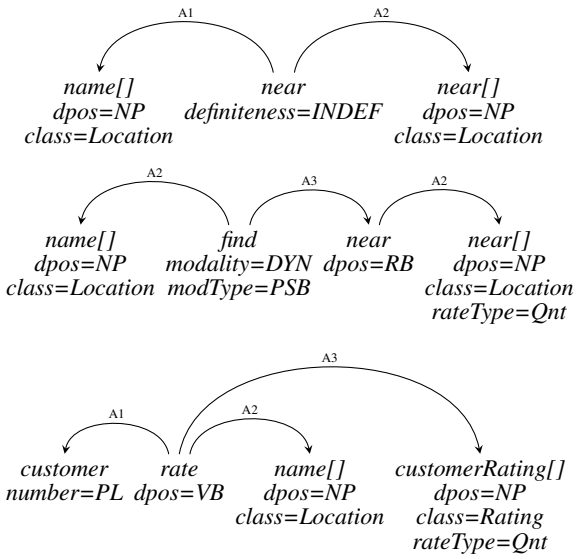


Figure 1: Sample PredArg templates: *near* (top, middle), and *customerRating* (bottom)

property and its object value type, namely quantitative or qualitative in the case of the *customerRating[]* and *priceRange[]* properties, and positive or negative in the case of the *familyFriendly[]* property, as depending on the combination, different predicate-argument templates apply.

From the resulting set of matching templates, and following a uniform distribution selection process, a predicate-argument template is chosen for each input entry triple. The population of the subject value, as afore-explained, is straightforward and corresponds to the value of the *name[]* property. For the object values, an additional step of pre-processing takes is required in order to normalize the "yes"/"no" values of the *familyFriendly[]* property and to convert the quantitative *priceRange[]* values (e.g. *less than £20*) into qualitative ones (e.g. low-priced) adopted in this implementation.

## 4 Aggregation of PredArg structures

Feeding the PredArg structures as such to the FORGe generator would render each property as an independent sentence. In order to group triples into complex sentences, a graph-transduction module is applied that performs aggregation in two steps.

On the one hand, generic rules look for shared pairs of predicate and subject argument in the populated templates, and introduce coordinations or quasi-coordinations between the two objects as

in:

*[Blue Spice]$_S$ is located$_P$ [in the city center]$_{O1}$ [near Rainbow Cafe]$_{O2}$.)*

Other generic rules check if an argument of a predicate appears further down in the ordered list of PredArg structures. If so, the PredArg structures are merged by fusing the common argument; during linguistic generation, this results in the introduction of post-nomimal modifiers such as relative and participial clauses or appositions; e.g.:

*The Cricketers$_S$, which serves$_{P2}$ [Italian food]$_{O2}$, has [a customer rating]$_{P1}$ of 1 out of 5$_{O1}$).*

On the other hand, rules specific to the E2E data have been implemented so as to aggregate objects that have not been aggregated by the generic rules; between the brackets, detail of the restrictions about co-occurring properties:

- eatType$_{P1}$ + priceRange$_{P2}$: *a cheap$_{O2}$ pub$_{O1}$*.
- eatType$_{P1}$ + familyFriendly$_{P2}$ (if no priceRange): *a family-friendly$_{O2}$ restaurant$_{O1}$*.
- *eatType$_{P1}$ + food$_{P2}$ (if no other property, or if one of the above): *an Italian$_{O2}$ restaurant$_{O1}$*.
- eatType$_{P1}$ + area$_{P2}$ (if none of the above): *a fast-food$_{O1}$ [in the riverside area]$_{O2}$*.
- eatType$_{P1}$ + near$_{P2}$ (if none of the above): *a pub$_{O1}$ [near Clowns]$_{O2}$*.
- eatType$_{P1}$ + customerRating$_{P2}$ (if none of the above): *a [high-rated]$_{O2}$ restaurant$_{O1}$*.
- area$_{P1}$ + near$_{P2}$ (if priceRange or if familyFriendly): *[in the riverside area]$_{O1}$ [near Clowns]$_{O2}$*.
- *food$_{P1}$ + priceRange$_{P2}$ (if no eatType): *cheap$_{O2}$ [Italian]$_{O1}$ food*.

The difference between the first two runs submitted lies in the deactivation of the two rules marked with a  for E2E_UPF_2, while E2E_UPF_1 makes use of all available rules. The two rules were added at a later stage in order to add variety in the proposed outputs, with the risk of producing less natural aggregations sometimes; as aforementioned, only E2E_UPF_1 was submitted as primary system.

Finally, another round of aggregation is run in order to bring together properties or values left alone after the application of the generic and specific rules. Referring expressions are introduced during linguistic generation.

## 5 Linguistic generation

The next and last step is the rendering of the aggregated PredArg structures into sentences. For this, we use the core FORGe grammars (Mille et al., 2017), but instead of the statistical linearization, we use a rule-based linearization component in order to have more control over the output quality. This part of the system follows the theoretical model of the Meaning-Text Theory (Mel'čuk, 1988), and performs the following actions: (i) syntacticization of predicate-argument graphs; (ii) introduction of function words; (iii) linearization and retrieval of surface forms.

### 5.1 Syntacticization

Before building the syntactic structures, we assign parts of speech to each node of the structure. This tagging is very rudimentary: rules check in a lexicon if a node label matches an entry and retrieve the part of speech from that entry. Since more than one part of speech is often possible for a given string, rules consult the context in the graph, if necessary, in order to take a decision.

Then, a top-down recursive syntacticization of the semantic graph takes place. It looks for the syntactic root of the sentence, and from there for its syntactic dependent(s), for the dependent(s) of the dependent(s), and so on. At this point, the structure contains all content words and the syntactic relations; it is a deep-syntactic structure in the sense of the Meaning-Text Theory.[5]

The rules in this level are organized in a cluster of three grammars. The purpose of the first two grammars is to identify the root of a syntactic tree. Verbs are the best candidates for being the root, and special attention is given to the number and complexity of dependents that a verb has. If no verb is available, adjectives, adverbs or nouns are

[5]For more details on the deep-syntactic structures and their relation with surface-syntactic structures in an NLP pipeline, see, e.g., (Ballesteros et al., 2015).

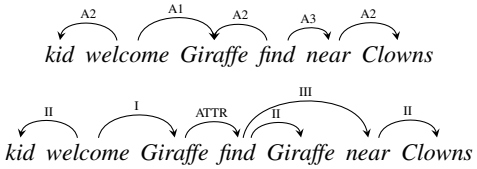


Figure 2: PredArg (top) and deep-syntactic (bottom) structures for the properties *name[Giraffe]*, *kids-Friendy[yes]*, *near[Clowns]*

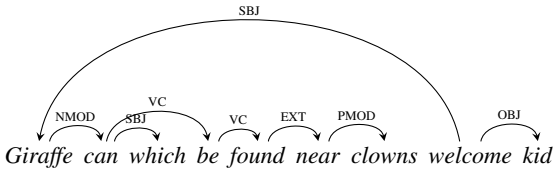


Figure 3: The surface-syntactic structure generated from the deep-syntactic structure in Figure2

selected and a support verb is introduced (as, e.g., in *Giraffe* $is_{SV}$ *near Clowns*).

In the following example, *welcome* is chosen as the root, and has two dependents *kid* and *Giraffe*. *Giraffe* is also the second argument of *find*, which is realized as an attribute of Giraffe, as the head of a relative clause. Figure 2 shows the aggregated PredArg structure and the corresponding deep-syntactic structure:

Note that the majority of the rules in the transduction grammars at this level are language-independent, since most of the language-specific information is encoded outside of the grammars, namely, in the lexical resources.

### 5.2 Introduction of functional words

The next step towards the realization of the sentence is the introduction of all idiosyncratic words and of a fine-grained (surface-)syntactic structure that gives enough information for linearizing and resolving agreements between the different words. For this task, we use valency (subcategorization) information extracted automatically from PropBank (Kingsbury and Palmer, 2002) and NomBank (Meyers et al., 2004): lexical units in the lexicon contain information about part of speech of dependents, bound prepositions, lexical functions, etc.; see (Mille and Wanner, 2015).

Functional words such as bound prepositions, determiners, and auxiliaries are introduced at this point. For instance, the predicate *find*, which is assigned a Dynamic modality of type *Possibility* in the PredArg template (see Figure 1) triggers the realization of a modal *can*. Moreover, the fact that there is no first argument below *find* means that the embedded clause needs to be realized in passive voice, its second argument (*Giraffe*) becoming the syntactic subject. Figure 3 shows a sample surface-syntactic structure.

Due to time constraints, we sometimes indicated which functional prepositions to use as a feature in the template in order to overrule an erroneous or missing entry in the lexicon.

During this transduction, anaphora are resolved, and personal pronouns are introduced in the tree (this includes possessive, relative and personal pronouns). Some syntactic post-processing rules fix ill-formed structures when possible.

### 5.3 Resolution of morpho-syntactic agreements

Every word must be assigned all necessary morphological information; some information comes from the deeper strata of the pipeline (as, e.g., verbal tense or finiteness), but some features come from some other elements of the tree. In English, for instance, verbs get their person and number from their syntactic subject, and determiners get their number and/or gender from their governing noun. In order to resolve these agreements, the rules for this transduction check the governor/dependent pairs, together with the syntactic relation that links them together.

Then the syntactic tree is linearized: first, governor/dependent pairs are ordered, and then dependents with respect to each other; during this step, punctuation marks are also introduced.

Finally, with the morpho-syntactic information at hand for each word, we just need to find the corresponding surface form. We match the tuple <lemma><POS><morpho-syntactic features> with an entry of a morphological dictionary and simply replace the triple by the surface form. In order to build such a dictionary, we analyzed a large amount of data and retrieved all possible combinations of lemma, part of speech and morphological features:

- nouns: number
  kid<NN><SG>=kid
  kid<NN><PL>=kids;

- verbs: finiteness (, tense, person, number)

find<VB><GER>=finding
find<VB><FIN><PRES><3><SG>=finds

If several surface forms are found for a combination of features, we select the most frequent one.

Continuing with the running example, the resulting sentence would be *He peeks at the black dog that barks*.

### 5.4 Post-processing

Last, a few post-processing rules are applied to make the output more readable: the first letter of a sentence is converted to upper case, final punctuation signs are added, underscores are replaced by spaces, and spaces before contracted elements ("'s" and "n't") are removed.

## 6 Full template-based generation

For E2E_UPF_3, our second primary submission, we implemented a very simple template-based generator: for each of the 108 combinations of properties, we manually (i) gathered between 2 and 9 different realizations in the training set, and (ii) replaced the values of the different properties by placeholders. Note that not all reference sentences can serve as templates, as certain verbalizations are bound to specific types of property values;[6] only suitable references have been singled out. In order to speed up the process, and to cover for cases in which no adequate template was found, we sometimes used as basis valid templates resulting from similar combinations of properties, and performed further manual edits if needed. See for instance the 5 possible templates for the combination *name[] eatType[] area[] near[]*:

AeatType[] called name[], can be found near the area[] next to near[].
There is a eatType[] name[] in the area[] area. It is near near[].
In the area[] area, there is a eatType[] called name[] near near[].
A eatType[] called name[] is located in the area[] area. It is near near[].
name[] is a eatType[] located in the area[] near near[].

For the same property, there are different templates in the case of (i) customer ratings, depending on if the value is qualitative (e.g. *high/low*) or quantitative (e.g. *1 out of 5*); and (ii) familyFriendly, depending on whether the value is *yes* or *no*. We compiled a total of 797 templates.

[6] E.g., the reference sentence "*There is a ¡priceRange¿ ¡eatType¿ that serves ¡food¿.*" is a valid candidate template for qualitative-only price ranges (*low-priced*, *moderately priced*, etc.).

The values for each property of the evaluation set simply substitutes the placeholders in each template.

## 7 Examples and error analysis

### 7.1 Correct outputs

In this section, we give a few sample outputs of both primary systems with no apparent issue.

**Rule-based System (E2E_UPF_1)**

1 Blue Spice, which is in riverside near Rainbow Vegetarian Café, is a non-family-friendly restaurant that offers Chinese food.

2 Blue Spice is a family-friendly restaurant that serves English cuisine. It is located in the city centre area and can be found near Rainbow Vegetarian Café.

3 Giraffe, which offers fast food, is a non-kid-friendly restaurant. It is in the city centre area by Rainbow Vegetarian Café.

4 The Cricketers, which customers rate 5 out of 5, is a non-family-friendly coffee shop. It can be found near Crowne Plaza Hotel.

5 The Cricketers, which has a high customer rating, is an average-priced restaurant that serves Chinese cuisine. It is located in the city centre area and can be found near All Bar One. It is kid-friendly.

**Template-based System (E2E_UPF_3)**

1 Blue Spice serves English food that is kids-friendly. It is in the city centre area near Rainbow Vegetarian Caf.

2 French food is available in a non children-friendly restaurant called Giraffe, near Raja Indian Cuisine in the riverside area.

3 The Cricketers is a cheap restaurant that serves Chinese food. It is not children-friendly and has an average customer rating. It is located in the city centre area near All Bar One.

4 The Phoenix, located near Express by Holiday Inn in the city centre area, is a not kid-friendly restaurant. The Phoenix serves cheap Indian food and has an average customer rating.

### 7.2 Unnatural/Erroneous outputs

I this section, we give a brief overview of the problems we found.

**Rule-based System (E2E_UPF_1)**

With the rule-based system, some outputs miss a comma, which makes the final text unnatural [1] (8 occurrences in the evaluation set), or a determiner *the* [2]. In other cases, the fluency is disrupted, with sentences that remain not aggregated [3], or some descriptive relative clauses [4].

1 The restaurant Blue Spice is located in the city centre area. It serves Chinese cuisine. Blue Spice which can be found near Rainbow Vegetarian Café welcomes kids.

2 The coffee shop Blue Spice is located in city centre.

3 Moderately priced The Cricketers is a restaurant offering Chinese food. It has a high customer rating. It is in riverside near All Bar One. It is not child-friendly.

4 The restaurant The Cricketers is by Crowne Plaza Hotel. Customers rate The Cricketers, which welcomes children, 5 out of 5.

### Template-based System (E2E_UPF_3)

The template-based system produced more incorrect outputs than the rule-based system. The main reason for this is a confusion when sending the output files out for evaluation; unfortunately, we submitted a trial version of the output, in which the values of the *priceRange* and *food* properties were inserted erroneously: in [1], *priced* is missing (*moderately priced*), in [2] it should be a qualitative rating (e.g. *mid-priced*) instead of a quantitative one (*More than £30*), while in [3] the word *food* is omitted (*English food*).

1 There is a 3 out of 5 stars moderate family-friendly restaurant The Cricketers near All Bar One in the city centre area. It offers Chinese food.

2 More than £30 Chinese food for adults can be found at The Cricketers restaurant, near All Bar One in the riverside area. High ratings.

3 For a child-friendly, high-rated restaurant serving English, try The Cricketers, in the city centre area, near All Bar One.

## 8 Results and discussion

Table 2 summarizes the results of the automatic and human evaluations:

- E2E_UPF_1 is the primary rule-based system, identical to the one submitted to WebNLG except for a few specific aggregation rules (see Section 4);
- E2E_UPF_2 is a variant of UPF_1 with different aggregation rules (see Section 4);
- E2E_UPF_3 is the other primary system, which uses handcrafted sentences with placeholders.

In the automatic evaluation, UPF_3 consistently outperforms the rule-based systems, but not for the human one. This is consistent with the fact that UPF_3 was made from sentences taken directly from the training set, hence more similar to what can be found in the evaluation set than the ones generated by the other two systems. The rule-based systems UPF_1 and UPF_2 achieve comparable scores, which again is expected since only a few rules are different. UPF_1, which has the final version of the aggregation rules, seems to perform slightly better, which is possibly due to the fact that the few additional rules introduce more variety in the UPF_1 structures, as in human-generated utterances.

| | **E2E_ UPF_1** | **E2E_ UPF_2** | **E2E_ UPF_3** | **Avg.** |
|---|---|---|---|---|
| BLEU | 42.07 | 41.13 | 45.99 | 58.13 |
| NIST | 6.51 | 6.33 | 7.11 | 7.62 |
| METEOR | 0.37 | 0.37 | 0.39 | 0.42 |
| ROUGE_L | 0.54 | 0.56 | 0.56 | 0.64 |
| CIDEr | 1.31 | 1.25 | 1.56 | 1.89 |
| Position Q | 2 (10-14) | N/A | 3 (15-16) | N/A |
| Position N | 4 (18-19) | N/A | 5 (20-21) | N/A |

Table 2: Results of the task with automatic metrics (top) and human evaluations (bottom; Q = Quality, N = Naturalness)

In the general classification, the three submitted systems are far below average for all automatic metrics. This is expected in this kind of task, in which data-driven systems are designed to output text that is as close to the training data as possible, which is not the case for rule-based systems for instance.

For the human evaluation, 21 primary systems were evaluated according to the quality and naturalness of the generated sentences, as defined in the following:[7]

*Quality is defined as an overall quality of the utterance, in terms of its grammatical correctness, fluency, adequacy and other important factors. When collecting quality ratings, system outputs were presented to crowd workers together with the corresponding meaning representation.*

*Naturalness is defined the extent to which the utterance could have been produced by a native speaker. When collecting naturalness ratings, system outputs were presented to crowd workers without the corresponding meaning representation.*

*If used in a real-life NLG system, quality would be considered the primary measure.*

Human evaluators compared systems outputs and ranked them according to these criteria; this intermediate ranking has been used to compute the

[7] http://www.macs.hw.ac.uk/InteractionLab/E2E/#results

final ranking, which contains five different clusters for each criterion (the first cluster contains the best system(s) according to one criterion). UPF_1 was ranked in the $2^{nd}$ and $4^{th}$ clusters for Quality and Naturalness respectively, and UPF_3 in the $3^{rd}$ and $5^{th}$. The last two rows of Table 2 also show the range in which the systems belong (within parenthesis).

For UPF_1, the results of the human evaluation are significantly better than the automatic one: although UPF_1 does not manage to often match the other systems (17 outputs are ranked higher) in terms of Naturalness, it is able to compete with the statistical systems for the Quality metric.

The negative human evaluation of UPF_3 can be easily explained by the amount of erroneous sentences (see Section 7), which unfortunately did not allow us to use this output as a reference, as planned initially.

## 9 Conclusions and future work

As shown in Section 7 and the human evaluation, our generator in a data-to-text context is able to produce good quality sentences, but it seems like it currently does not reach the level of naturalness achieved by statistical systems.

In the future, the aggregation and generation of referring expressions modules will be refined; for instance, now we only use string matching to generate pronouns, and do not control the possible ambiguities about the antecedent. We will also work on making the input predicate-argument more flexible during the population phase.

## Acknowledgments

This work has been partially funded by the European Commission under the contracts H2020-645012-RIA, H2020-700024-RIA, and H2020-700475-RIA. We would also like to thank the task organizers and the anonymous reviewers for their valuable feedback.

## References

Miguel Ballesteros, Bernd Bohnet, Simon Mille, and Leo Wanner. 2015. Data-driven deep-syntactic dependency parsing. *Natural Language Engineering* pages 1–36.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*. Las Palmas, Canary Islands, Spain, pages 1989–1993.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An interim report. In *Proceedings of the Workshop on Frontiers in Corpus Annotation, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*. Boston, MA, USA, pages 24–31.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. Forge at semeval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 917–920. http://www.aclweb.org/anthology/S17-2158.

Simon Mille and Leo Wanner. 2015. Towards large-coverage detailed lexical resources for data-to-text generation. In *Proceedings of the First International Workshop on Data-to-text Generation*. Edinburgh, Scotland.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Saarbrücken, Germany. ArXiv:1706.09254. https://arxiv.org/abs/1706.09254.