

Scalable Micro-planned Generation of Discourse from Structured Data

Anirban Laha*
Mila, Université de Montréal
anirbanlaha@gmail.com

Parag Jain*
School of Informatics,
University of Edinburgh
parag.jain@ed.ac.uk

Abhijit Mishra*
IBM Research
abhijimi@in.ibm.com

Karthik Sankaranarayanan
IBM Research
kartsank@in.ibm.com

We present a framework for generating natural language description from structured data such as tables; the problem comes under the category of data-to-text natural language generation (NLG). Modern data-to-text NLG systems typically employ end-to-end statistical and neural architectures that learn from a limited amount of task-specific labeled data, and therefore, exhibit limited scalability, domain-adaptability, and interpretability. Unlike these systems, ours is a modular, pipeline-based approach, and does not require task-specific parallel data. It rather relies on monolingual corpora and basic off-the-shelf NLP tools. This makes our system more scalable and easily adaptable to newer domains.

Our system employs a 3-staged pipeline that: (i) converts entries in the structured data to canonical form, (ii) generates simple sentences for each atomic entry in the canonicalized representation, and (iii) combines the sentences to produce a coherent, fluent and adequate paragraph description through sentence compounding and co-reference replacement modules. Experiments on a benchmark mixed-domain dataset curated for paragraph description from tables reveals the superiority of our system over existing data-to-text approaches. We also demonstrate the robustness of our system in accepting other popular datasets covering diverse data types such as Knowledge Graphs and Key-Value maps.

1. Introduction

Structured data, such as tables, knowledge graphs, or dictionaries containing key-value pairs are popular data representation mechanisms used in a wide variety of industries to capture domain-specific knowledge. As examples, (1) in the *finance* domain, tabular data representing the financial performance of companies, (2) in *healthcare*, information about chemical composition of drugs, patient records *etc.*, (3) in *retail*, inventory records of products and their features, are few among many other manifestations of structured data. Various AI-based human-machine interaction applications such as question-answering or dialog involve retrieving information from such structured data for their end goals. A key component in such applications deals with Natural

* The first three authors have equally contributed to the work

Existing Structured data-to-text systems : While **WEBNLG Model** is a system trained on WEBNLG training data (Gardent et al. 2017), **WIKIBIO Model** is a system trained on WIKIBIO training data (Lebret, Grangier, and Auli 2016). As expected, the performance of both these models are good only on the dataset they are trained on, showing lacking of adaptability across domains.

Language Generation (NLG) from the aforementioned structured data representations, known as the *data-to-text* problem. Another important use-case of this problem is *story-telling* from data, as in automatic report generation.

2

Further, since existing systems are designed as task-specific solutions, they tend to jointly learn both *content selection* from the input (what to say?) and the *surface realization* or language generation (how to say?). This is often undesirable as the former, which decides “what is interesting” in the input, can be highly domain-specific. For example, what weather parameters (temperature, wind-chill) are influential versus what body parameters (heart-rate, body temperature) are anomalous are heavily dependent on the domain at hand, such as weather or healthcare respectively. Whereas the latter part of language generation may not be as much domain dependent and can, thus, be designed in a reusable and scalable way. Therefore, it would be easier to develop scalable systems for language generation independently than developing systems that jointly learn to perform both content selection and generation.

In this article, we propose a general purpose, unsupervised approach to language generation from structured data; our approach works at the linguistic level using word and sub-word level structures. The system is primarily designed for taking a structured table with variable schema as input and producing a coherent paragraph description pertaining to the facts in the table. However, it can also work with other structured data formats such as graphs and key-value pairs (in the form of JSONs) as input. Multiple experiments show the efficacy of our approach on different datasets having varying input formats without being trained on any of these datasets. By design, the system is unsupervised and scalable, *i.e.* it assumes no labeled corpus and only considers monolingual, unlabeled corpora and WordNet during development, which are inexpensive and relatively easy to obtain.

In the proposed approach, the generation of description from structured data happens in three stages, *viz.* (1) **canonicalization**, where the input is converted to a standard canonical representation in the form of tuples, (2) **simple language generation**, where each canonical form extracted from the input is converted into a simple sentence, and (3) **discourse synthesis and language enrichment**, where simple sentences are merged together to produce complex and more natural sentences. The first stage is essential to handle variable schema and different formats. The second stage gleans morphological, lexical, and syntactic constituents from the canonical tuples, and stitches them into simple sentences. The third stage applies sentence compounding and co-reference replacement on the previously produced simple sentences to generate a fluent and adequate description. For the development of these modules, at most a monolingual corpus, WordNet, and three basic off-the-shelf NLP tools namely, part-of-speech tagger, dependency parser and named entity recognizer are needed.

To test our system, we first curate a multi-domain benchmark dataset (referred henceforth as **WIKITABLEPARA**) that contains tables and corresponding manually written paragraph descriptions; and to the best of our knowledge, such a dataset does not exist. Our experimental results on this dataset demonstrate the superiority of our system over the existing *data-to-text* systems. Our framework can also be extended to different schema and datatypes. To prove this, we perform additional experiments on two datasets representing various domains and input-types, only using their test splits: (i) WIKIBIO (Lebret, Grangier, and Auli 2016), representing key-value pairs, and (ii) WEBNLG (Gardent et al. 2017), representing knowledge graphs. Additionally, for the sake of completeness, we extend our experiments and test our system’s performance on existing *data-to-text* NLG datasets (for the task of tuple to text generation). We demonstrate that even though our system does not undergo training on any of these datasets, it nevertheless delivers promising performance on their test splits. The key contributions of this article are as summarized below:

- We propose a general purpose, unsupervised, scalable system for generation of descriptions from structured tables with variable schema and diverse formats.
- Our system employs a modular approach enabling interpretability, as the output of each stage in our pipeline is in a human-understandable textual form.

- We release a dataset called **WIKITABLEPARA** containing WikiTables and their descriptions for further research. Additionally, we also release data gathered for modules for sentence realization from tuples (refer *Secs. 4*, and *6*, useful for general purpose tuple/set to sequence tasks. The dataset and code for our experiments are available at <https://github.com/parajain/structscribe>.

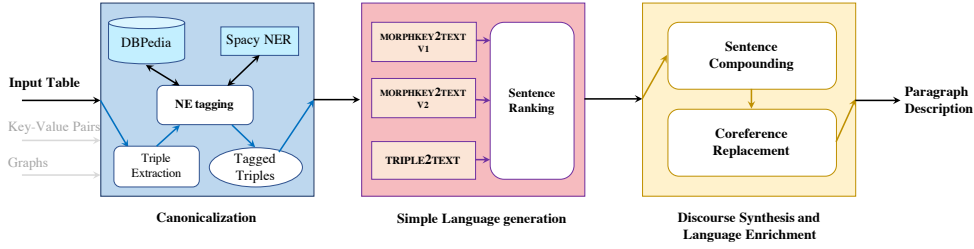
We would like to remind our readers that our system is unsupervised as it does not require parallel corpora containing structured data such as tables at the source side end and natural language description at the target side. Manually constructing such labeled data can be more demanding than some of the well-known language generation tasks (such as summarization and translation) because of the variability of the source structure and the non-natural association between the source and target sides. Our system does not require such parallel data and divides the problem into sub-problems. It requires only simpler data forms that can be curated from unlabeled sources.

We would also like to point that an ideal description generation system would require understanding the pragmatic aspects of the structure under consideration. Incorporating pragmatic knowledge still remains an open problem in the domain of NLG, and our system’s capability towards handling pragmatics is rather limited. As the state-of-the-art progresses, we believe that a modular approach such as the one proposed can be upgraded appropriately.

2. Central Challenges and Our Solution

This section summarizes the key challenges in description generation from structured data.

- **Variable Schema:** Tables can have variable number of rows and columns. Moreover, the central theme around which the description should revolve can vary. For example, two tables can contain column-headers [*Company Name*, *Location*], yet the topic of the description can be the *companies* or the *locations* of various companies. Also, two tables having column-headers [*PlayersName*, *Rank*] and [*Rank*, *PlayersName*] represent the same data but may be handled differently by existing-methods that rely on ordered-sequential inputs.
- **Variation in Presentation of Information:** The headers of tables typically capture information that is crucial for generation. However, presentations of headers can considerably vary for similar tables. For example, two similar tables can have column-headers like [*Player*, *Country*] and [*Player Name*, *Played for Country*], where the headers in the first table are single-word nouns but the first header of the second table is a *noun-phrase* and the second header is *verb-phrase*. It is also possible that the headers share different inter-relationships. Nouns such as [*Company*, *CEO*] should represent the fact that CEO is a part of the company, whereas entities in headers [*temperature*, *humidity*] are independent of each other.
- **Domain Influence:** It is known that changing the domain of the input has adverse effects on end-to-end generators, primarily due to differences in vocabulary (*e.g.*, the word “tranquilizer” in healthcare data may not be found in tourism data).
- **Natural Discourse Generation:** Table descriptions in the form of discourse (paragraphs) should contain a natural flow with a mixture of simple, compound, complex sentences. Repetition of entities should also be replaced by appropriate co-referents. In short, the paragraphs should be fluent, adequate and coherent.

**Figure 1**

Our proposed 3-staged modular architecture for description generation from structured data

End-to-end neural systems mentioned in the previous sections suffer from all the above challenges. According to [Gardent et al. \(2017\)](#), these systems tend to overfit the data they are trained on, “*generating domain specific, often strongly stereotyped text*” (eg. weather forecast or game commentator reports). Rather than learning the semantic relations between data and text, these systems are heavily influenced by the style of the text, the domain vocabulary, input format of the data and co-occurrence patterns. As per [Wiseman, Shieber, and Rush \(2017\)](#), “*Even with recent ideas of copying and reconstruction, there is a significant gap between neural models and template-based systems, highlighting the challenges in data-to-text generation*”. Our system is designed to address the challenges to some extent through a three-staged pipeline, namely, (a) **canonicalization**, (b) **simple language generation**, and (c) **discourse synthesis and language enrichment**. In the first stage, the input is converted to a standard canonical representation in the form of tuples. In the second stage, each canonical form extracted from the input is converted to simple sentences. In the final stage, the simple sentences are combined to produce coherent descriptions. The overall architecture is presented in *Fig. 1*.

Note that our pipeline is designed to work with tables which do not have a hierarchy amongst its column headers and row headers. We believe that tables of such kind can be normalized as a pre-processing step and then fed to our system. To handle this pre-processing is beyond the scope of the current work. We discuss our central idea in the following sections.

3. Canonicalization of Structured Data

Our goal is to generate descriptions from structured data which can appear in various formats. For this, it is essential to convert the data to a canonical form which can be handled by our generation stages. Though our main focus is to process data in tabular form, the converter is designed to handle other input formats as well, as discussed below.

3.1 Input Formats

1. *Table*: Tables are data organized in rows and columns. We consider single-level row and column headers with no hierarchy. A table row can be interpreted as an n -ary relation. Currently, we simplify table row representation as a collection of binary relations (or triples).
2. *Graph*: Knowledge Graphs have entities represented as nodes while edges denote relations between entities. Here we consider only binary relations. A knowledge graph can be translated as a collection of binary relations or triples.

3. *JSON*: This is data organized in the form of a dictionary of key-value pairs. We limit ourselves to single-level key-value pairs where the keys and values are literals. A pair of key-value pairs are converted to a triple by concatenating the value term of the first key-value pair with the second key-value pair.

3.2 Canonical Form and Canonicalization

For our system to be able to handle various formats listed above, we need to convert them to a standard format easily recognizable by our system. Moreover, it is required that the generation step can be trained without involving labeled parallel data so that they can be used in various domains where only monolingual corpora is available. Keeping this in mind, we arrived at a canonical form consisting of triples made of binary relations among two entities types. For example, consider the triple : $\langle \textit{Albert Einstein} ; \textit{birth place} ; \textit{Ulm, Germany} \rangle$. The entity tags for named entities ‘Albert Einstein’ and ‘Ulm, Germany’ are PERSON and GPE respectively. This leads to a canonicalized triple form as the following

$\langle \textbf{PERSON birth place GPE} \rangle$

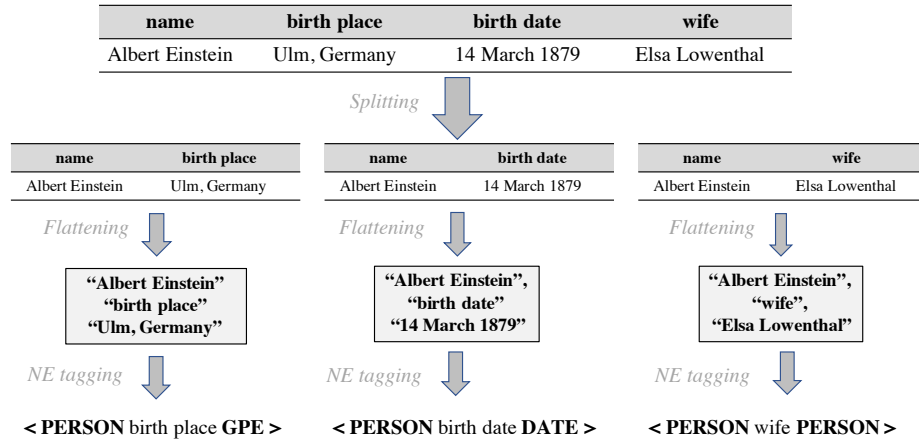
For tabular inputs, extraction of tuples require the following assumption to be followed.

- The column-headers of the table should be considered as the list of keywords that decide the structure of the sentences to be generated. In case the table is centered around row headers (*i.e.*, row headers contain maximum generic information about the table), the table has to be transposed first.
- One column header is considered as the *primary key*, around which the theme of the generated output revolves. For simplicity, we chose the first column-header of the tables in our dataset to be the primary key.

For each table, the table is first broken into a set of subtables containing 1-row and 2-columns, as shown in *Fig. 2*. The first columns of the subtables represent the primary-key of the table. For a table containing M rows and N columns (excluding headers), a total number of $M \times (N - 1)$ subtables are thus produced. The subtables are then flattened to produce a triple by dropping the primary key header and concatenating the entries of the subtables, as shown in *Fig. 2*. This produces standard entity-relationship triples $\langle e_1, r, e_2 \rangle$ where e_1 , and e_2 are entities that are entries and r is the relationship, which is captured by the column header.

The entities e_1 , and e_2 are tagged using an NER-tagger, which assigns domain-independent place-holder tags such as **PERSON** and **GPE** for persons and geographical regions respectively. For tagging we use Spacy (spacy.io) NER tagger, an off-the-shelf tagger that performs reasonably well even on words and phrases. We also employ a DBpedia lookup¹ based on exact string matching in situations where NER is unable to recognize the named entity. String matching is done with either the URI-labels or the anchor-texts referring to the URI to find out the relevant tag. This is helpful in detection of peculiar multi-word named entities like ‘The Silence of the Lambs’ which will not be recognized by Spacy due to lack of context. All DBpedia classes have been manually mapped to 18 Spacy NER types. As a fallback mechanism, any entity not recognized through DBpedia lookup is assigned with **UNK** tag. This process produces somewhat domain independent canonical representations from the tables as seen in *Fig. 2*. The NER tags

¹ Refer <https://github.com/dbpedia/lookup>

**Figure 2**

Example of extraction of canonical triples from tabular inputs

and the corresponding original entries are carried forward and remain available for use in stages 2 and 3. At stage 2, these tags are replaced with the original entries to form proper sentences. The tags and the original entries are also used for language enrichment in stage 3.

Unlike tables, for input types like knowledge graphs and key-value pairs, extraction of canonical triples is straightforward. Knowledge graphs typically follow the triple form with nodes representing entities and edges representing relations. Similarly, a pair of key-value entries can be flattened and a triple can be extracted. All these formats, thus, can be standardized to a collection of canonical triples with NE tags acting as placeholders.

In the following section, we describe how a simple sentence can be extracted from each canonical triple. A collection of canonical triples obtained from a table (or other input types) will produce a collection of simple sentences, which is compounded to form a coherent description.

4. Simple Language Generation

The simple language generation module takes each canonical triple and generates a simple sentence in natural language. For instance, the triple $\langle \text{PERSON birth place GPE} \rangle$ will be translated to a simple sentential form like the following:

$\langle \text{PERSON was born in GPE} \rangle$

This will finally be replaced with the original entities to produce a simple sentence as follows: *Albert Einstein was born in Ulm, Germany*. The canonical triple set in Fig. 2 should produce the following (or similar) simple sentences (refer as set 1):

Albert Einstein was born in Ulm, Germany
Albert Einstein has birthday on 14 March 1879
Elsa Lowenthal is the wife of Albert Einstein
(1)

This is achieved by the following steps : (1) **Preprocessing** - which transforms the canonical triple to a modified canonical triple, (2) **TextGen** - which converts the modified canonical triple to

a simple sentential form like $\langle \text{PERSON was born in GPE} \rangle$, (3) **Postprocessing** - which replaces back the original entities to produce a simple sentence like *Albert Einstein was born in Ulm, Germany*, and lastly, (4) **Ranking** - which selects the best sentence produced in step 3 when multiple variants of TextGen are run in parallel. The details of these steps are shared below.

4.1 Preprocessing

It is possible that the canonical triples will contain words that cannot be easily converted to a sentence form without additional explicit knowledge. For example, it may not be easy to transform the vanilla triple $\langle \text{PERSON game Badminton} \rangle$ to a syntactically correct sentence $\langle \text{PERSON plays Badminton} \rangle$.

To convert the relation term into a verb phrase we employ a pre-processing step. The step requires two resources to be available - (1) WordNet and (2) Generic Word embeddings, at least covering the default vocabulary of the language (English). We use the 300-dimensional *glove* embeddings for this purpose (Pennington, Socher, and Manning 2014).

The preprocessing step covers the following two scenarios:

1. **Relation term is a single-word term:** In this case, the word is lemmatized and the root form is looked up in a verb lexicon pre-extracted from WordNet. If the look up succeeds, the lemma form is retained in the modified triple. Otherwise, the top N verbs² that are closest to the word are extracted using *glove* vector based *cosine similarity*. For example, through this technique, for the original word “game”, which is not a verb, related verbs such as “match” and “play” can be extracted. The verb “play” will be the most suitable one for generating a sentence later. The most suitable verb is decided as follows. For each extracted verb v related to the original word o , the *synsets* for v and o are extracted from WordNet. The glosses and examples for each synset of o are extracted from WordNet and combined to form a textual representation (F_o). Similarly, the textual representation (F_v) considering the glosses and examples of synsets of v is formed. The *degree of co-occurrence* of words v and o is computed using the normalized counts of co-occurrences of v and o in F_o and F_v . The candidate verb having the highest *degree of co-occurrence* is selected as the most appropriate verb. Through this, the word “play” would be selected as the most appropriate verb for the word “game”, as both words will co-occur in the glosses and examples of synsets of both “game” and “play”.
2. **Relation term is a multi-word term:** The relation term, in this case, would contain both content (*i.e.* non-stopwords) and function words (*i.e.* stopwords). Examples of multi-word terms are “country played for” and “number of reviews”. When such terms are encountered, the main verb in the phrase is extracted through *part-of-speech* (POS) tagging. If a verb is present, the phrase is altered by moving the noun phrase preceding the verb to the end of the phrase. So, the phrase “country played for”, through this heuristic, would be transformed to “played for country”. This is based on the assumption that in tabular forms, noun phrases that convey an *action* are actually a transformed version of a verb phrase.

² N is set to 10 in our setup.

The above preprocessing techniques modify the input triple which we refer to as *modified canonical triple*. This step is useful for the TRIPLE2TEXT generation step as discussed next.

4.2 TextGen

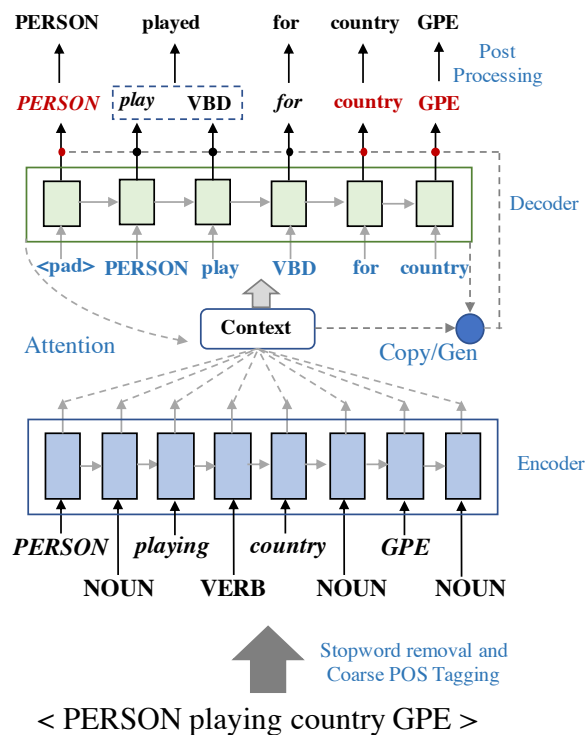
The objective of this step is to generate simple and syntactically correct sentences from the (modified) canonical triples. We propose three ways to generate sentential forms as elaborated below. All the below mentioned ways are different alternatives to generate a simple sentential form, hence they can be executed in parallel.

4.2.1 TRIPLE2TEXT. This module is the simplest and is developed using a `seq2seq` (Klein et al. 2017) network, which is trained on the curated TRIPLE2TEXT dataset (refer Sec. 6 Dataset 3). The dataset consists of triples curated from various sources of knowledge bases extracted from open web-scale text dumps using popular information extraction techniques (such as Banko et al. (2007); Schmitz et al. (2012)). Additionally, existing resources such as Yago Ontology (Suchanek, Kasneci, and Weikum 2007) and VerbNet (Schuler 2005) are employed. The criteria for constructing triples and simple sentence pairs (used as target for training `seq2seq`) are different for different resources. We should point to our readers that no annotation was needed for creation of this dataset as the simple sentences were constructed by concatenation of elements in the triples (discussed in Sec. 6 Dataset 3). Only this variant of generation requires a modified canonical triple, obtained using the preprocessing step mentioned above. The other variants can work with the canonical triple without such modification.

4.2.2 MORPHKEY2TEXT (v1 and v2). The conversion of any canonical triple to a sentence demands the following linguistic operations:

1. determining the appropriate morphological form for the words/phrase in the canonical triple, especially the relation word/phrase (*e.g.*, transforming the word “play” to “played” or “plays”).
2. determining the articles and prepositions necessary to construct the sentences (*e.g.*, transforming “play” to “plays for”).
3. adding appropriate auxiliary verbs when necessary. This is needed especially for passive forms (*e.g.*, transforming “location” to “is located at” by adding the auxiliary verb “is”).

Ideally, any module designed for canonical-triple to sentence translation should dynamically select a subset of the above operations based on the contextual clues present in the input. To this, we propose the MORPHKEY2TEXT module, a variant of `seq2seq` network empowered with attention and copy mechanisms. Fig. 3 shows a working example of the MORPHKEY2TEXT system. We skip explaining the well-known `seq2seq` framework for brevity. As input, the module takes a processed version of the canonical triple in which (a) NE tags are retained (b) Stopwords are removed if they appear in the relation terms in the canonical triples and (c) The coarse POS tags for both the NE tags and words are appended to the input sequence. The module is expected to generate a sequence of words along with the fine-grained POS tags (in PENN tagset format) for the verbs appearing in their *lemma* form. The rationale behind such an input-output design is that, **dealing with the lemma forms at the target side and incorporating additional linguistic signals in terms of POS should enable the system to apply appropriate changes at morphological and lexical levels**. This will, in turn, help address the problem of lexical and morphological data-sparsity across domains better. As seen in Fig. 3, the canonical

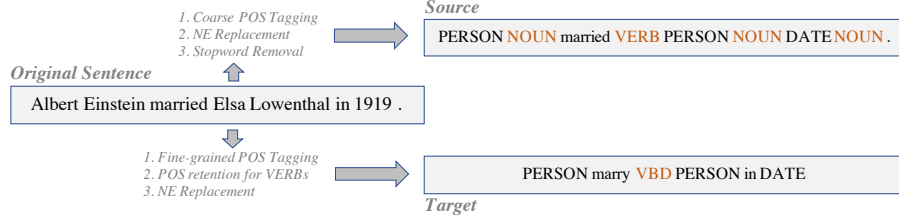
**Figure 3**

Example demonstrating the working of the MORPHKEY2TEXT TextGen system. Generated words shown in red color are produced via copy operation and those in black color are produced via generation operation.

triple $\langle \text{PERSON playing country GPE} \rangle$, is first transformed into a list of content words and their corresponding coarse-grained POS tags. During generation, the input key-word and POS “playing VERB” are translated to “play VBD” and the output is post-processed to produce the word “played”. As the system has to deal with lemma forms and NE and POS tags at both input and output sides, it allows the system to just *copy* input words, which makes the system robust across domains.

Preparing training data for the MORPHKEY2TEXT design requires only a monolingual corpus and a few general purpose NLP tools and resources such as POS tagger, NE Tagger, and WordNet. A large number of simple sentences extracted from web-scale text dumps (such as Wikipedia) are first collected. The sentences are then POS tagged and the named entities are replaced with NE tags. Stopwords (function words) such as articles and prepositions are dropped from the sentences by looking up in a stopwords lexicon. Since the POS tagger produces fine-grained POS-tags, the tags are converted to coarse POS tags using a predefined mapping. This produces the *source* (input) side of the training example. As of *target* (output), the named entities in the original sentences are replaced with NE tags, the other words are *lemmatized* using WordNet lemmatizer, and the fine-grained POS tags of the words are augmented if the lemma form is not the same as the base form. Fig. 4 illustrates construction of a training example from unlabeled data.

We implement two different variants of the MORPHKEY2TEXT system. The MORPHKEY2TEXT V1 module is trained based on the MORPHKEY2TEXT dataset (version v1) that

**Figure 4**

Extraction of a single training instance from an unlabeled sentence for the MORPHKEY2TEXT TextGen system.

was created from monolingual corpora (explained in *Sec. 6* Dataset 2). The MORPHKEY2TEXT v2 is trained on a different version (v2) of the MORPHKEY2TEXT dataset (details in *Sec. 6* Dataset 2).

4.3 Post-processing

This step restores the original entities from the input by replacing the tagged forms generated from the step above. Additionally, if possessive nouns are detected in the sentence, apostrophes are added to such nouns. Possessives are checked using the following heuristic - *if the POS tag for the word following the first entity is not a verb, the word is a potential possessive candidate*. Postprocessing is applied to each of the competing modules enlisted in the above step.

The above variants TRIPLE2TEXT, MORPHKEY2TEXT v1 and MORPHKEY2TEXT v2 can run in parallel to produce different translations of the canonical triple. Out of these, the best produced sentence are selected by the ranking step mentioned below.

4.4 Scoring and Ranking

To select the most appropriate output from the TextGen systems discussed earlier, a **ranker** is employed; it sorts the sentence based on a composite score as given below:

$$score(i, s) = f(s) \times g(i, s) \quad (2)$$

where i and s represent the canonical triple and generated sentence. Functions $f(\cdot)$ and $g(\cdot)$ represent the *fluency* (grammaticality) of the output sentence and *adequacy* (factual overlap between input and output). The *fluency* function f is defined as follows:

$$f(s) = LM(s) = LM(w_1, w_2, \dots, w_N) \quad (3)$$

where for a sentence of N words w_1, w_2, \dots, w_N , the LM, an N -gram language model, returns the likelihood of the sentence. For this, a 5-gram general purpose language model is built using Wikipedia dump and KenLM (Heafield 2011). The *adequacy* function g is defined as:

$$g(i, s) = \frac{\#co\text{-occurring words in } i \text{ and } s}{\#words \text{ in } i} \quad (4)$$

Before applying the ranker, we employ heuristics to filter incomplete and un-natural sentences. Sentences without verbs or entities and sentences that are disproportionately larger or

Algorithm 1 COMPOUND (s_1, s_2, D)

```

1:  $e_{11}, rvp_1, e_{12} \leftarrow \text{SplitIntoTuple}(\text{Sentence } s_1, D)$ 
2:  $e_{21}, rvp_2, e_{22} \leftarrow \text{SplitIntoTuple}(\text{Sentence } s_2, D)$ 
3: REM The function SplitIntoTuple splits a sentence  $s$  into a triple by considering the entities and their corresponding
   types, the mapping is provided by dictionary  $D$ .
4: if  $e_{11} = e_{21} \ \& \ rvp_1 = rvp_2$  then
5:   REM e.g., Jordan played basketball and football.
6:   return " $e_{11} \ rvp_1 \ e_{12}$  and  $e_{22}$ "
7: else if  $e_{11} = e_{21}$  then
8:   REM e.g., Jordan played basketball and represented U.S.A.
9:   return " $e_{11} \ rvp_1 \ e_{12}$  and  $rvp_2 \ e_{22}$ "
10: else if  $e_{12} = e_{22} \ \& \ rvp_1 = rvp_2$  then
11:   REM e.g., Jordan and Kurt played basketball.
12:   return " $e_{11}$  and  $e_{21} \ rvp_1 \ e_{22}$ "
13: else if  $e_{12} = e_{22}$  then
14:   REM e.g., Jordan loved and Kurt hated basketball.
15:   return " $e_{11} \ rvp_1$  and  $e_{21} \ rvp_2 \ e_{22}$ "
16: else if  $e_{12} = e_{21} \ \& \ \text{TypeOf}(e_{12}) = \text{PERSON}$  then
17:   REM e.g., Jordan married Prieto who is a model from Cuba.
18:   return " $e_{11} \ rvp_1 \ e_{12}$  who  $rvp_2 \ e_{22}$ "
19: else if  $e_{12} = e_{21}$  then
20:   REM e.g., Jordan played basketball which featured in movie Space Jam.
21:   return " $e_{11} \ rvp_1 \ e_{12}$  which  $rvp_2 \ e_{22}$ "
22: end if

```

smaller than the input are discarded. Once the ranker produces the best simple sentence per input triple, the simple sentences are then combined into a coherent paragraph as explained below.

5. Discourse Synthesis and Language Enrichment

In this section, we discuss how to combine the collection of generated simple sentences set 1 from Sec. 4 to produce a paragraph by merging sentences as shown below:

Albert Einstein was born in Ulm, Germany and has birthday on 14 March 1879. Elsa Lowenthal is the wife of Albert Einstein.

The above paragraph is produced by a *sentence compounding module* followed by a *coreference replacement module* to produce the final coherent paragraph:

Albert Einstein was born in Ulm, Germany and has birthday on 14 March 1879. Elsa Lowenthal is the wife of him.

5.1 Sentence Compounding

This module takes a pair of simple sentences and produces a compound or complex sentence. Every simple sentence is split into a $\langle e_1, rvp, e_2 \rangle$ form where e_1 and e_2 are entities that appear in the input and rvp is the relation verb phrase. For a pair of sentences, if both sentences share the same first entity e_1 or both have the same second entity e_2 , the compounded version can be obtained by ‘AND’-ing of the relation phrases. In cases where the second entity of one matches the first entity of the following sentence, then a clausal pattern can be created by adding “*who*” or “*which*”. In all other cases, the sentences can be merged by ‘AND’-ing both the sentences. Alg. 1 elaborates on this heuristic. This module can also generate different variations of paragraphs based on different combinations of sentences.

5.2 Coreference Replacement

To enhance paragraph coherence, it is often desirable to replace entities that repeat within or across consecutive sentences with appropriate coreferents. For this, we employ a heuristic that replaces repeating entities with pronominal anaphora.

If an entity is encountered twice in a sentence or appears in consecutive sentences, it is marked as a potential candidate for replacement. The number and gender of the entity are decided using POS tags and an off-the-shelf Gender Predictor module. The module is a CNN-based classifier that trains on person names gathered from various websites. The entity’s role is determined based on whether it appears to the left of the verb (*i.e.*, *Agent*) or to the right (*Object*). Based on the gender, number, role and possessives, the pronouns (he/she/their/him/his *etc.*) are selected, and they replace the entity. We ensure that we replace only one entity in a sentence to avoid incoherent construction due to multiple replacements in close proximity.

We remind our reader about *Fig. 1* that presents an overview of our system. Due to its modular nature, our system enjoys interpretability; each stage in the pipeline is conditioned on the output of the previous stage. Moreover, all the modules, in principle, can adapt to newer domains. The datasets used for training do not have any domain-specific characteristics and thus these modules can work well across various domains as will be seen in the “experiments” section. The whole pipeline can be developed without any parallel corpora of structured table to text. Any data used for training any individual module can be curated from monolingual corpora. The subsequent section discusses such datasets in detail.

6. Datasets

The section discusses three datasets; Dataset 1 contains tables from various domains and their summaries and can be used for benchmarking any table descriptor generator. Dataset 2 and 3 are developed to train our TextGen modules (*Sec. 4*). These datasets can be downloaded for academic use from <https://github.com/parajain/structscribe>. We also release the code and resources to create similar datasets in a larger scale.

6.1 Dataset 1: Descriptions from WikiTable (WIKITABLEPARA)

We prepare a benchmark dataset for multi-sentence description generation from tables. For gathering input tables, we rely on the WIKITABLE dataset ([Pasupat and Liang 2015](#)), which is a repository of more than 2000 tables. Most of the tables still suffer from the following issues: (a) they do not provide enough context information, as they were originally a part of a Wikipedia page, (b) they are concatenations of multiple tables, and (c) they contain noisy entries. After filtering such tables, we extract 171 tables. Four reference descriptions in the form of paragraphs were manually generated. The average number of sentences for each description in each reference is 12 and the average number of words is between 740-780 respectively. The descriptions revolve around one column of the table, which acts as the primary-key.

6.2 Dataset 2: Morphological Variation based Keywords-to-Text (MORPHKEY2TEXT)

This is created from monolingual corpora released by ([Thorne et al. 2018](#)), which is a processed version of Wikipedia dump. We create the first version of the dataset following the technique discussed in *Sec. 4.2.2, paragraph. 3*, using POS- and NE taggers.

The second version v2 is slightly different in the sense that it employs a higher-recall oriented entity tagging mechanism with the help of POS tags and dependency parse trees of sentences. This is necessary as there are entities such as “*A Song of Ice and Fire*”, which will not

Table 2
Statistics for Datasets 2 and 3

Dataset	#Instances	Avg. #words in target	#Target vocabulary
TRIPLE2TEXT	33188424	3.45	5594
MORPHKEY2TEXT-V1	9481470	9.74	876153
MORPHKEY2TEXT-V2	9346617	8.51	477302

be recognized by the NE tagger used to create v1. Such multi-word entities can be detected by a simple heuristic which looks for a sequence of proper nouns (in this case ‘Song’, ‘Ice’ and ‘Fire’) surrounded by stop-words but do not include any punctuation. Moreover, it should not have any verb marked as `root` by the dependency parser. Through this technique, it is also possible to handle cases where an entity such as “*Tony Blair*” gets detected as two entity tags PERSON and UNK by popular NE taggers such as Spacy, instead of single entity-tag PERSON.

6.3 Dataset 3: Knowledge Base Triples to Text (TRIPLE2TEXT)

For this, a large number of triples and corresponding sentential forms are gathered from the following resources. (i) *Yago Ontology*: 6198617 parallel triples and sentences extracted from Yago (Suchanek, Kasneci, and Weikum 2007). Our improvised NER, discussed in Sec. 3 is used for getting tags for entities in the triples. (ii) *OpenIE on WikiData*: 53066988 parallel triples and sentences synthesized from relations from Reverb Clueweb (Banko et al. 2007) and all possible combinations of NE Tags. (iii) *VerbNet*: 149760 parallel triples and sentences synthesized from verbs (in the first person singular form) from VerbNet (Schuler 2005) and possible combinations of NE Tags. For all the knowledge resources considered for this dataset, concatenation of the elements in the triples yielded simple sentences, hence there was no manual effort needed for creation of this dataset.

Various statistics for dataset 2 and 3 are presented in Table. 2. For training the TextGen systems, the datasets were randomly divided into train, valid and test splits of 80%:10%:10%.

7. Experiments

The simple language generator in Sec. 4 require training `seq2seq` networks using the MORPHKEY2TEXT (v1 and v2) and the TRIPLE2TEXT datasets. For this we use the OPENNMT framework in PyTorch, using the default hyperparameter settings. The best epoch model is chosen based on accuracy on the validation split of the above datasets. Once these modules are trained, they are used in inference mode in our pipeline.

Through experiments, we show the efficacy of our proposed system on WIKITABLEPARA and other public *data-to-text* benchmark datasets even though it is not trained on those datasets. Additionally, we also assess the generalizability of our and other existing end-to-end systems in unseen domains. We use BLEU-4, METEOR, ROUGE-L and *Skip-Thoughts* based Cosine Similarity (denoted as STSim) as the evaluation metrics³. We also perform a human evaluation study, where a held-out portion of the test data is evaluated by linguists who assign scores to the generated descriptions pertaining to *fluency*, *adequacy* and *coherence*. Mainly, we try to answer

³ <https://github.com/Maluuba/nlg-eval>

the following research questions through our empirical study:

1. **Can other existing end-to-end systems adapt to unseen domains?** For this, we consider two pre-trained representative models: (a) WIKIBIOMODEL (Nema et al. 2018) - A neural model trained on the WIKIBIO dataset (Lebret, Grangier, and Auli 2016), and (b) WEBNLGMODEL⁴ - A *seq2seq* baseline trained on the WEBNLG dataset (Colin et al. 2016; Gardent et al. 2017). These models are tested on the WIKITABLEPARA dataset which is not restricted to any particular domain. Additionally, they are also tested on two popular tuple-to-text datasets such as E2E (Novikova, Dušek, and Rieser 2017) and WIKITABLETEXT (Bao et al. 2018). Thus, the performance of the existing systems can be assessed on wide variety of domains which may not have been present in the datasets used for developing the systems.
2. **How well our system adapts to new domains?** We evaluate our proposed system also on the table-to-descriptions WIKITABLEPARA benchmark dataset to contrast the performance with the above pretrained models. Additionally, we also assess our system on *related (table-to-text summarization)* datasets: (1) WEBNLG, (2) WIKIBIO, (3) WIKITABLETEXT, and (4) E2E. The WIKITABLETEXT dataset, like ours, is also derived from WikiTables. However, it contains only tabular-rows and their summary in one sentence. The generation objective becomes different from ours, as it does not require paragraph level operations such as compounding and coreference resolution. Therefore, for brevity, we only report our system’s performance on the dataset without further analysis.
3. **How interpretable is our approach?** By leveraging the modularity of our system, we would analyze the usefulness of major components in the proposed system and perform error analysis.

7.1 Experimental Setup

We now discuss how the various systems are configured for evaluation on multiple datasets.

- **PROPOSED SYSTEM :** Our proposed system is already designed to work with the format of the WIKITABLEPARA dataset. Each table in the dataset is converted to $M \times (N - 1)$ canonical triples leading to the output table description (refer Sec. 3). To test our system for other input types such as *Knowledge Graphs* and *Key-Value* dictionaries, we use the WEBNLG and WIKIBIO datasets respectively. From WIKIBIO dataset, JSONs containing N Key-Value pairs $\langle key1:value1, key2:value2, \dots, keyN:valueN \rangle$ are converted to $N - 1$ triples. Each triple is in the form $\langle value1, keyI, valueI \rangle$, where $I \neq 1$. It is assumed that the first key is the primary key and typically contains names and other keywords for identifying the original wikipedia infobox. For WEBNLG dataset, the triples in a group are directly used by our system to produce the output. For the WIKITABLETEXT dataset, which contains one tuple per instance, each input is converted into $N - 1$, triples, in similar manner as the WIKITABLEPARA dataset. For the E2E dataset, each instance already is in triple-to-text form, and is used as it is.

⁴ <http://webnl.g.loria.fr/pages/baseline.html>

- **WEBNLGMODEL** : The WEBNLGMODEL is designed to be trained and tested on the WEBNLG dataset. An already trained WEBNLGMODEL model (similar to the one by [Gardent et al. \(2017\)](#)) is evaluated on WIKITABLEPARA and WIKIBIO datasets. For WIKITABLEPARA dataset, we convert every table to $M \times (N - 1)$ triples. For each triple, the model infers a sentence and sentences for all the triples representing a table are concatenated to produce a paragraph description. For the WIKIBIO dataset, each JSON is converted to $N - 1$ triples for N key-value pairs, which are then passed to the model for final output. Tuples in WIKITABLETEXT dataset are converted to $N - 1$ triples and instances in E2E dataset, which are already in triple-to-text are used directly without any transformation.
- **WIKIBIOMODEL** : The WIKIBIOMODEL is designed to get trained and tested on the WIKIBIO dataset that contains Key-Value pairs at the input side and summaries at the output. An already trained model (similar to the one by [Nema et al. \(2018\)](#)) is evaluated on WIKITABLEPARA and WEBNLG datasets. For the WIKITABLEPARA dataset, we convert every table to $M \times (N - 1)$ jsons in WIKIBIO format. Each JSON contains a pair of Key-Value pairs, where the first Key-Value pair always represents the primary-key and its corresponding entry in the table (hence, $N - 1$ JSONs are produced). The inferred sentences for all $M \times (N - 1)$ jsons from the model are concatenated to produce the required paragraph description. For the WEBNLG dataset, each triple is converted to a JSON of a pair of Key-Value pairs. A triple $\langle e_1, r, e_2 \rangle$ is converted to a JSON format of $\{default_key : e_1, r : e_2\}$ (the default key is set to “name”). For each instance in the WEBNLG dataset, sentences are inferred for all the triples belonging to the instance, and they are concatenated to produce the final output. Inputs from WIKITABLETEXT and E2E datasets are converted to JSON as explained above.

Please note that both WIKIBIOMODEL and WEBNLGMODEL are capable of processing single and multi-tuple inputs. For our dataset, we try giving these models inputs in both single and multi-tuple format. In single-tuple input mode, the model processes one triple at a time and produces a sentence; the sentences are concatenated to produce paragraphs. In multi-tuple mode, all triples extracted from a single row of the table are simultaneously passed to the model as input. The model variants with subscript “ M ” represent these cases in the result tables. For the above evaluations, only the test splits for WIKIBIO and WEBNLG datasets are used, whereas there is no train:test split for the WIKITABLEPARA dataset (the entire dataset is used for evaluation). The results for these are summarized in [Tables 3 and 4](#).

7.1.1 Ablation Study. Apart from comparing our system with the existing ones, we also try to understand how different stages of our pipeline contribute to the overall performance. For such an ablation study, we prepare the different variants of the system based on the following two scenarios and compare their performance against that of the complete system.

- Instead of using the ensemble (RANKER), each participating TextGen systems *viz.* TRIPLE2TEXT, MORPHKEY2TEXT V1 and MORPHKEY2TEXT V2 are treated as separate system. The intention is to show the advantage of using an ensemble of generators and the ranking mechanism.
- Language enrichment modules such as compounding and coreference replacement modules are removed both individually and together. Simple sentences are just concatenated to produce the table descriptions. The intention is to test our

Table 3
Various models on WIKITABLEPARA dataset

System	BLEU	METEOR	ROUGE-L	STSim
WIKIBIOMODEL	0.0	15.5	14.8	64.1
WEBNLGMODEL	7.9	24.8	27.9	78.2
PROPOSED	33.3	39.7	64.1	86.5

hypothesis that removing such modules will make the generated paragraphs somewhat incoherent and deviate from constructs produced by humans, thereby resulting in a reduced system performance.

8. Results and Discussion

Table 3 illustrates how the various pretrained models fare on the WIKITABLEPARA benchmark dataset compared to our proposed system. We observe that the end-to-end WEBNLGMODEL does better than WIKIBIOMODEL. However, our proposed system clearly gives the best performance, demonstrating the capability of generalizing in unseen domains and structured data in a more complex form such as multi-row and multi-column table.

It may be argued that although the proposed model is not trained on parallel data, it takes advantage from the fact that the textual resources used for development come from the same sources as the test data (*i.e.*, Wikipedia). Thus, the better performance can be attributed to having a better vocabulary coverage (covering more entities, verbs, nouns *etc.*) which WEBNLGMODEL and WIKIBIOMODEL are deprived of. This is, however, not true because of two reasons: (1) the WIKIBIO and WEBNLG datasets use information from Wikipedia (in the form of Infoboxes and DBpedia entries), or (2) use pre-trained Glove embeddings (Pennington, Socher, and Manning 2014), which offers a much richer vocabulary than what is considered in our setting. Hence, it is evident that the performance of these baseline systems is low on WIKITABLEPARA dataset not because of vocabulary *unseenness* but for the very fact that these systems are rigid with respect to the language patterns seen in the data they are trained on.

It may also seem unfair to compare standalone systems like WIKIBIOMODEL and WEBNLGMODEL with an ensemble model like ours, as the latter may have infused more knowledge because of the inclusion of supporting modules. Again, this is not entirely true. The WIKIBIOMODEL under consideration is more sophisticated than a vanilla sequence-to-sequence model and employs attention mechanisms at various levels to handle intricacies in content selection and language generation (Nema et al. 2018). The WEBNLGMODEL employs various normalization and post-processing steps to adapt to newer domains and language patterns. In sum, these models are capable of handling nuances in data-to-text generation and, hence, deem fit for comparison.

Table 4 shows the performance of our proposed system on the test splits of various datasets (including the whole WIKITABLEPARA dataset). The performance measures (especially the STSim metric) indicate that our system can be used as it is for other input types coming from diverse domains. Despite the fact that the WIKITABLETEXT, WEBNLG and WIKIBIO datasets are summarization datasets and are not designed for complete description generation, our system still performs reasonably well, without having been trained on any of these datasets. It is clearly observed that the existing end-to-end models such as WEBNLGMODEL and WIKIBIOMODEL

Table 4Evaluation of all models across datasets (domains). Suffix M represents multi-tuple input.

Model	Dataset	BLEU	METEOR	ROUGE-L	STSim
Proposed	WEBNLG	24.8	34.9	52.0	82.6
	WIKITABLETEXT	12.9	33.6	37.1	73.2
	WIKIBIO	2.5	17.6	19.3	72.9
	E2E	6.6	27.1	29.2	71.1
	WIKITABLEPARA	33.3	39.7	64.1	86.5
WIKIBIOMODEL	WEBNLG	2.8	16.9	26.4	72.1
	WIKITABLETEXT	1.3	10.5	21.5	66.5
	E2E	1.3	9.0	22.7	61.6
	WIKITABLEPARA	0.0	15.5	14.8	64.1
	WIKITABLEPARA _M	0.0	10.3	13.7	65.8
WEBNLGMODEL	WIKITABLETEXT	3.6	16.5	25.2	68.9
	WIKIBIO	1.6	9.3	18.6	69.4
	E2E	2.1	13.2	19.0	66.0
	WIKITABLEPARA	7.9	24.8	27.9	78.2
	WIKITABLEPARA _M	0.5	20.0	26.1	75.6

exhibit inferior cross-domain performance compared to our system⁵. For example, our system attains BLEU scores of 24.8 and 2.5 on WEBNLG and WIKIBIO datasets respectively, whereas, the WIKIBIOMODEL performs with a BLEU score of 2.8 (with a reduction of 89%) on the WEBNLG dataset and the WEBNLGMODEL performs with a BLEU score of 1.6 (with a reduction of 36%) on WIKIBIO dataset. For other datasets such as WIKITABLETEXT and E2E, on which none of the proposed or comparison systems are trained on, our system’s performance is significantly better than the comparison systems. For the E2E dataset, we observe that our system’s outputs convey similar semantics as the reference texts but have considerable syntactic differences. For examples, the triples **⟨Taste of Cambridge eat type restaurant⟩**, and **⟨Taste of Cambridge customer rating 3 out of 5⟩** are translated to “*Taste of Cambridge is an eat type of restaurant and has a customer rating of 3 out of 5.*” by one of our model variant, but the reference text is “*Taste of Cambridge is a restaurant with a customer rating of 3 out of 5.*” This may have affected the BLEU scores; the METEOR and semantic relatedness scores are still better.

We performed ablation on our proposed system at multiple levels; *Table 5* shows the performance of individual simple language generation systems and also the performance of the ranker module. The results suggest that ranker indeed improves the performance of the system. To measure the effectiveness of our proposed sentence compounding and coreference replacement modules, we replaced these modules with a simple sentence concatenation module. As observed in the same table, the performance of the system degrades compared to when compounding and coreference replacement modules are individually used. Best results are obtained when both the modules are activated. One of the possible reasons is that a simple sentence concatenation results in generated paragraphs having more redundant occurrences of entity terms and phrases,

⁵ Please note that WIKIBIOMODEL trained on WIKIBIO dataset (in-domain) would have considerably higher evaluation scores (refer [Nema et al. \(2018\)](#)); the same holds for the WEBNLGMODEL ([Gardent et al. 2017](#)). Since our objective is to highlight cross-domain performance (where testing is done on datasets different from training data), the in-domain results are not discussed for brevity.

Table 5

Ablation study: Performance of individual TextGen systems with the ensemble system enabled with RANKER. Here, MKT denotes MORPHKEY2TEXT, CP refers to the Compounding Module and CR means Coreference Replacement. The symbols ‘+’ and ‘−’ signify “with” and “without” respectively.

		RANKER	MKTv1	MKTv2	TRIPLE2TEXT
BLEU	-CP-CR	17.7	16.2	14.9	20.3
	+CP-CR	30.1	29.7	27.9	30.3
	-CP+CR	29.6	29.3	28	27.8
	+CP+CR	33.3	30.6	29.4	30.5
METEOR	-CP-CR	33.1	33.4	32.6	31.6
	+CP-CR	38.8	39.3	38.4	36.7
	-CP+CR	37.1	37	37	34.8
	+CP+CR	39.7	38.1	38.1	35.4
ROUGE-L	-CP-CR	50.2	51	49.1	50.6
	+CP-CR	61.8	62.3	60.7	61
	-CP+CR	59.2	59	58.7	58.4
	+CP+CR	64.1	63.9	62.2	62.2
STSim	-CP-CR	44.1	40.2	40.2	57.8
	+CP-CR	85.3	85.6	85.2	83.3
	-CP+CR	82.3	82	82	79.8
	+CP+CR	86.5	85.9	85.9	83.9

Table 6

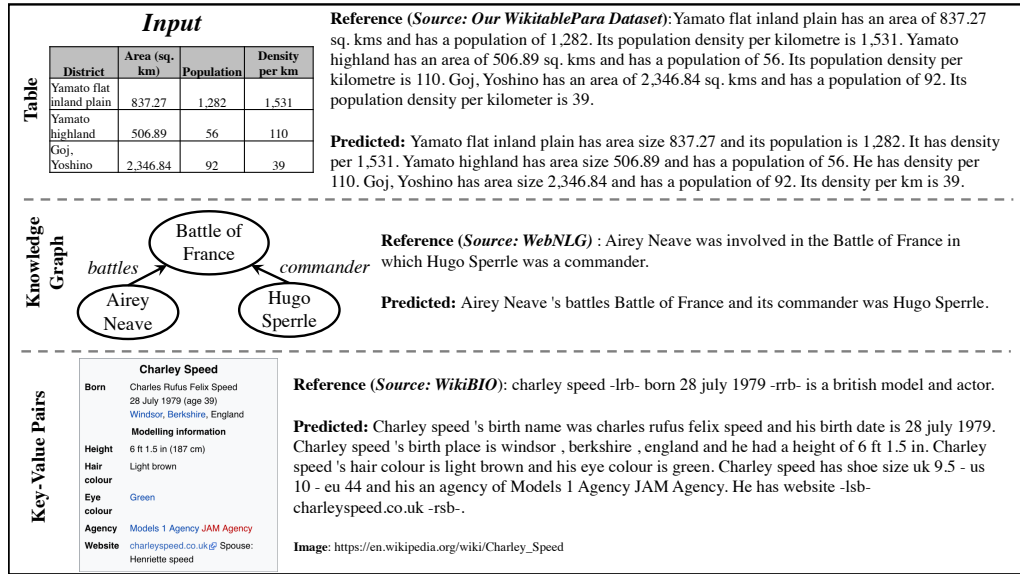
Human evaluation using 50 samples from the WIKITABLEPARA dataset. The fluency, adequacy and coherence scores are averaged across evaluators and instances. Evaluator correlation is the Pearson Correlation which shows the agreement between evaluators.

System	Fluency	Adequacy	Coherence
WIKIBIOMODEL	1.44	1.24	1.08
WEBNLGMODEL	2.04	2.05	1.66
PROPOSED	3.29	4.20	3.72
GOLD-standard	4.53	4.78	4.59
<i>Evaluator Correlation</i>	0.74	0.80	0.76

which all of the evaluation metrics tend to penalize heavily. Overall, this study goes to show that the enrichment modules indeed play an important role, especially when it comes to paragraph description generation.

8.1 Human Evaluation

Since quantitative evaluation metrics such as BLEU and Skip-thought similarity are known to have limited capabilities in judging sentences that are correct but different from the gold-standard reference, we perform a human evaluation study. For this, the first 50 instances from the WIKITABLEPARA dataset were selected. For each instance, the table, the reference paragraph, and outputs from our proposed system, WIKIBIO and WEBNLG models were shuffled and shown to four linguists. They were instructed to assign three scores related to fluency, adequacy

**Figure 5**

Examples of generated descriptions by our proposed system on different datasets.

and coherence of the generated and gold-standard paragraphs. The minimum and maximum scores for each category are 1 and 5 respectively. *Table. 6* reports the evaluation results. While it was expected that the gold-standard output would get maximum average scores in all aspects, the scores for our proposed systems are quite superior to the existing systems and are also sometimes close to those for the gold standard paragraphs. This shows that a modular approach like ours can be effective for generating tabular descriptions. Moreover, the average Pearson Correlation coefficient values for each scores across systems and evaluator-pairs are high, showing a strong inter-evaluator agreement.

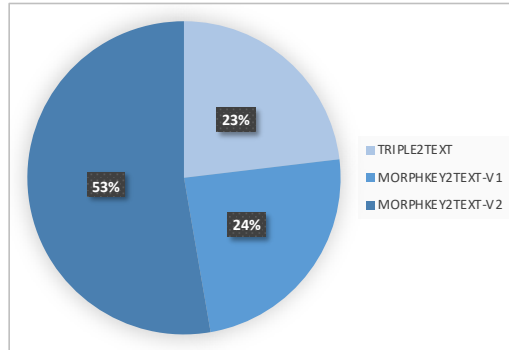
On manual inspection of the descriptions generated by our system across datasets (some examples are shown in *Fig. 5*), we find that our system gives a promising performance in addition to the quantitative evaluation metrics mentioned before.

8.2 Effectiveness of the Individual Modules

We also examine if, for TextGen, using an ensemble of generators followed by a ranking mechanism was effective. We intend to study if all the participating systems were chosen by the ranker for a significant number of examples. *Fig. 6* shows the percentage of the times the output of the three TextGen systems were selected by the ranker. As we can see, all systems are significantly involved in producing the correct output in the test data. However, the TRIPLE2TEXT system is selected fewer number of times than the other two systems. This is a positive result as the TRIPLE2TEXT system requires data obtained from specific resources such as OpenIE, and Yago as opposed to the MORPHKEY2TEXT systems that require just a monolingual corpus.

8.3 Error Analysis

Since our system is modular, we could inspect the intermediate outputs of different stages and perform error-analysis. We categorize the errors into the following:

**Figure 6**

Contribution of each TextGen system in term of percentage of times the output of these systems were selected by the ranker for the WIKITABLEPARA dataset.

- Error in Tagging of Entities:** One of the crucial steps in the canonicalization stage is tagging the table entries. Our modified NE taggers sometimes fail to tag entities primarily because of lack of context. For example, the original triple in our dataset $\langle \textit{Chinese Taipei}, \textit{gold medals won}, 1 \rangle$ is converted to a triple $\langle \textit{UNK}, \textit{gold medals won}, \textit{CARDINAL} \rangle$. Because of the wrong NE-tagging of the entity Chinese Taipei, the text generation stage in the pipeline did not get enough context and failed to produce a fluent output as shown below,

Chinese Taipei's gold medals have been won by 0.

This error affected all the subsequent stages. While it is hard to resolve this with existing NLP techniques, maintaining and incrementally building gazetteers of domain specific entities for look-up based tagging can be a temporary solution.

- Error in the TextGen:** We observe that all the TextGen systems, discussed in *Sec. 4.2* are prone to syntactic errors, which are mostly of types *subject-verb disagreement*, *noun-number disagreement*, article and preposition errors. An example of such an erroneous output is shown below:

Republican 's active voters is 13,916. Republican was inactive in voters 5,342

We believe such errors can be avoided by adding more training examples, judiciously prepared from large scale monolingual data from different domains.

- Error in Ranking:** This error impacts the performance of our system the most. We consistently observe that even though one of the individual systems is able to produce fluent and adequate output, it is not selected by the ranker module. In the hindsight, scorers based on simple language models and content-overlaps (*Eq. 2*) are not able to capture diverse syntactic and semantic representations of the same context (*e.g.*, passive forms, reordering of words). Moreover, language models are known to capture N-gram collocations better than the overall context of the sentences, and tend to penalize grammatically correct sentences more than the incorrect sentences that have more *likely* collocations of N-grams. Furthermore, longer sentences are penalized more by the language model than shorter ones. To put this into perspective, consider the following example from our dataset. For the

input triple, $\langle \textit{Bischofsheim}, \textit{building type}, \textit{Station building} \rangle$, the output from the TextGen systems are as follows:

TRIPLE2TEXT: *Bischofsheim has building type Station Building.*

MORPHKEY2TEXT-V1: *Bischofsheim's building is a type of Station Building.*

MORPHKEY2TEXT-V2: *Bischofsheim is a building type of Station Building.*

The ranker unfortunately selects an imperfect output produced by the MORPHKEY2TEXT-V2 system. We believe that the presence of highly probable bigrams such as *building type* and *type of* would have bolstered the language model score and, eventually the overall score. A possible solution to overcome this would be to train neural knowledge language models (Ahn et al. 2016) that not only considers contextual history but also factual correctness of the generated text. Gathering more monolingual data for training such models may help as well.

- **Error in Coreference Determination:** Error in coreference determination happens due to two reasons : (a) The entities are incorrectly tagged (e.g., a *PERSON* is mis-tagged as *ORG*, leading to a wrong pronominal anaphora.), and (b) The gender of the entity is incorrectly classified (e.g., Esther Ndiema's nationality is Kenya and *his* rank is 5). While improving the tagger is important for this and the overall system, the gender detector could be improved through more training data and better tuning of hyperparameters. The current module does have limitations due to the fact that it is based on a very small number of heuristics and relies on data-driven POS-taggers and gender predictors, which may not provide accurate information about the number and gender of the mentions. For example, for an entity "Mariya Papulov", even though POS tagger and the canonicalized entity tag (*PERSON*) help determine the number of the coreference correctly, the gender predictor assigns the gender tag as male. This results in a wrong co-reference assignment.

A deeper issue with the sentence enrichment modules is that they are agnostic of the sentence order. If the TextGen systems do not provide sentences in an appropriate order to these modules, the cohesiveness of the generated paragraph is compromised. For example, the output from our system "*Melania Corradini played for Italy and was on the run of distance 54.72 KMs. She had the rank of 5.*" provides a less natural feel than "*Melania Corradini played for Italy and had the rank of 5. She was on the run of distance 54.72 KMs*". This clearly calls for a technique to determine the optimal order of sentences to ensure more naturalness in the output.

We would also like to point out that language enrichment through simple concatenation and heuristic based replacement is a rudimentary solution. Better solutions for compounding and producing coherent paragraphs may involve syntactic analysis and restructuring of sentences (Narayan et al. 2017) and discourse aware coherent generation (Narayan et al. 2017; Kibble and Power 2004; Bosselut et al. 2018).

9. Related Work

Data-to-text NLG has received a lot of attention in recent times, especially due to the increasing demands of such systems for industrial applications. Several such systems are based on rule-based, modular statistical and hybrid approaches and are summarized by Nema et al. (2018). Recently, end-to-end neural generation systems have been preferred over others. Some of the most recent ones are based on the WIKIBIO dataset (Lebret, Grangier, and Auli 2016), a dataset tailor-made for summarization of structured data in the form of key-value pairs. Such systems

include the ones by [Lebret, Grangier, and Auli \(2016\)](#), who use conditional language model with copy mechanism for generation, [Liu et al. \(2017\)](#) who propose a dual attention Seq2Seq model, [Nema et al. \(2018\)](#) who employ gated orthogonalization along with dual attention, and [Bao et al. \(2018\)](#) who introduce a flexible copying mechanism that selectively replicates contents from the table in the output sequence. Other systems revolve around popular datasets such as WEATHERGOV dataset ([Liang, Jordan, and Klein 2009](#); [Jain et al. 2018](#)), ROBOCUP dataset ([Chen and Mooney 2008](#)), ROTOWIRE and SBNATION ([Wiseman, Shieber, and Rush 2017](#)), and WEBNLG dataset ([Gardent et al. 2017](#)). Recently [Bao et al. \(2018\)](#) and ([Novikova, Dušek, and Rieser 2017](#)) have introduced a new dataset for table/tuple to text generation and both supervised and unsupervised systems ([Fevry and Phang 2018](#)) have been proposed and evaluated against these datasets. The objective of creating such datasets and systems is, however, entirely different from ours. For instance, [Bao et al. \(2018\)](#)’s objective is to generate natural language summary for a **region** of the table such as a row, whereas, we intend to translate the complete table into paragraph descriptions. This requires additional discourse level operations (such as sentence compounding and coreference insertion). The dataset also contains tabular rows at the input side and summaries at the output side. Since the objective is summarization of a tabular region, a fraction of the entries are dropped and not explained, unlike ours that aims to translate the complete table into natural language form.

It is worth noting that recent works for keyword to question generation ([Reddy et al. 2017](#)), and set-to-sequence generation ([Vinyals, Bengio, and Kudlur 2016](#)) can also act as building blocks for generation from structured data. However, none of these works consider the morphological and linguistic variations of output words as we consider for simple language generation. Also, the work on neural knowledge language model ([Ahn et al. 2016](#)) incorporates facts towards a better language model, which is a different goal from ours as we attempt to describe the full table of facts in natural language in a coherent manner. However, irrespective of the generation paradigms we use, the bottom line remains the same - modular data driven approaches like ours can produce robust and scalable solutions for data-to-text NLG.

It is also worth noting that there exist well-studied information extraction (IE) techniques to obtain tuples from sentences which could be employed to prepare additional parallel training data for improving any data-to-text NLG solution. Here, the tuples can be used as source and the original (or preprocessed) sentences can be used as target for training supervised generators. Popular IE techniques include Open Information Extraction ([Banko et al. 2007](#)) and Open Language Learning for Information Extraction ([Schmitz et al. 2012](#)). These systems leverage POS taggers and dependency parsers to extract relation tuples and are in line with our method for data generation. However, they do not consider lexical and morphological aspects of the sentences considered as we did for MORPHKEY2TEXT. From the domain of relation extraction, works such as ([Mintz et al. 2009](#)), that extract training datasets for relation annotation using distant supervision techniques, can also be improvised and used in our setting.

10. Conclusion and Future Directions

We presented a modular framework for generating paragraph level natural language description from structured tabular data. We highlighted the challenges involved and contended why a modular data-driven architecture like ours could tackle them better as opposed to end-to-end neural systems. Our framework employs modules for obtaining standard representations of tables, generating simple sentences from them and finally combining the sentences to form coherent and fluent paragraphs. Since no benchmark dataset for evaluating discourse level description generation was available, we created one to evaluate our system. Our experiments on our dataset and various other *data-to-text* type datasets reveal that: (a) our system outperforms the existing ones in producing discourse level descriptions without undergoing end-to-end training on such

data, and (b) the system can realize good quality sentences for various other input data-types such as *knowledge graphs* in the form of tuples and key-value pairs. Furthermore, the modularity of the system allows us to interpret the system’s output better. In the future, we would like to incorporate additional modules into the system for tabular summarization. Extending the framework for multilingual tabular description generation is also on our agenda.

References

- Ahn, Sungjin, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.
- Banko, Michele, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.
- Bao, Junwei, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. *arXiv preprint arXiv:1805.11234*.
- Barzilay, Regina and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *EMNLP, HLT '05*, pages 331–338, Association for Computational Linguistics, Stroudsburg, PA, USA.
- Bosselut, Antoine, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *NAACL, HLT*, pages 173–184, Association for Computational Linguistics, New Orleans, Louisiana.
- Chen, David L and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*, pages 128–135, ACM.
- Colin, Emilie, Claire Gardent, Yassine Mrabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The webnlg challenge: Generating text from dbpedia data. In *INLG*, pages 163–167.
- Dale, Robert, Sabine Geldof, and Jean-Philippe Prost. 2003. Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 35–44, Australian Computer Society, Inc.
- Fevry, Thibault and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. *arXiv preprint arXiv:1809.02669*.
- Gardent, Claire, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *ACL*, pages 179–188.
- Heafield, Kenneth. 2011. Kenlm: Faster and smaller language model queries. In *Sixth Workshop on SMT*, pages 187–197, Association for Computational Linguistics.
- Jain, Parag, Anirban Laha, Karthik Sankaranarayanan, Preksha Nema, Mitesh M. Khapra, and Shreyas Shetty. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *NAACL-HLT*, pages 622–627, Association for Computational Linguistics.
- Kibble, Rodger and Richard Power. 2004. Optimizing referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.
- Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810.
- Konstas, Ioannis and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *EMNLP*, pages 1503–1514.
- Lebret, Rémi, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213, Association for Computational Linguistics, Austin, Texas.
- Liang, Percy, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99, Association for Computational Linguistics.
- Liu, Tianyu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724*.
- Mei, Hongyuan, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *NAACL, HLT*, pages 720–730, Association for Computational Linguistics, San Diego, California.
- Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*, pages 1003–1011, Association for Computational Linguistics.
- Narayan, Shashi, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *EMNLP*, pages 606–616, Association for Computational Linguistics.
- Nema, Preksha, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M. Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *NAACL-HLT*, pages

- 1539–1550, Association for Computational Linguistics.
- Novikova, Jekaterina, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *SIGdial*, pages 201–206, Association for Computational Linguistics.
- Pasupat, Panupong and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *IJCNLP*, pages 1470–1480, Association for Computational Linguistics.
- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.
- Reddy, Sathish, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *EACL*, pages 376–385, Association for Computational Linguistics, Valencia, Spain.
- Reiter, Ehud, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.
- Schmitz, Michael, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534, Association for Computational Linguistics.
- Schuler, Karin Kipper. 2005. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.
- Suchanek, Fabian M, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, ACM.
- Thorne, James, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819, Association for Computational Linguistics, New Orleans, Louisiana.
- Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *ICLR*.
- Wiseman, Sam, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *EMNLP*, pages 2253–2263, Association for Computational Linguistics.

—

—

—

—

—

—