Data2Text Studio: Automated Text Generation from Structured Data

Longxu Dou[1]*, **Guanghui Qin**[2]*, **Jinpeng Wang**[3], **Jin-Ge Yao**[3], **and Chin-Yew Lin**[3]
[1] Harbin Institute of Technology, dreamerdeo@hit.edu.cn
[2] Peking University, ghq@pku.edu.cn
[3] Microsoft Research Asia, {jinpwa, jinge.yao, cyl}@microsoft.com

Abstract

Data2Text Studio is a platform for automated text generation from structured data. It is equipped with a Semi-HMMs model to extract high-quality templates and corresponding trigger conditions from parallel data automatically, which improves the interactivity and interpretability of the generated text. In addition, several easy-to-use tools are provided for developers to edit templates of pre-trained models, and APIs are released for developers to call the pre-trained model to generate texts in third-party applications. We conduct experiments on ROTOWIRE datasets for template extraction and text generation. The results show that our model achieves improvements on both tasks.

1 Introduction

Data-to-text generation, i.e., a technology which takes structured data as input and produces text that adequately and fluently describes this data as output, has various applications on the generation of sports news (Chen and Mooney, 2008; Kim and Mooney, 2010; Mei et al., 2016; Wiseman et al., 2017), product descriptions (Wang et al., 2017), weather reports (Liang et al., 2009; Angeli et al., 2010; Mei et al., 2016) and short biographies (Lebret et al., 2016; Chisholm et al., 2017). In another scenario, it is possible albeit a little awkward for a virtual assistant like Microsoft Cortana to read out structured data when responding to users' queries. it is more user friendly for a virtual assistant to identify and read out the essential part of the structured data in natural language to make it easier to understand. In these cases, it is inefficient and expensive to generate texts using human writers, while an automatic text generation system would be helpful.

There are two main challenges for data-to-text generation systems: 1) Interactivity: For a developer, it should be able to customize the text generation model and control the generated texts. 2) Interpretability: the generated texts should be consistent with the structured data. For example, we can say "with a massive 8 GB of memory" for a laptop computer while "a massive 2 GB" is inappropriate. Rule-based approaches (Moore and Paris, 1993; Hovy, 1993; Reiter and Dale, 2000; Belz, 2007; Bouayad-Agha et al., 2011) encode domain knowledge into the generation system and then produce high-quality texts, while the construction of the system is expensive and heavily depends on domain experts. Statistical approaches are employed to reduce extensive development time by learning rules from historical data (Langkilde and Knight, 1998; Liang et al., 2009; Duboue and McKeown, 2003; Howald et al., 2013). However, statistical approaches are prone to generating texts with mistakes, because they don't know how to use specific phrases under various application conditions.

To address the second challenge, we propose a Semi-HMMs model to automatically extract templates and corresponding trigger conditions from parallel training data. Trigger conditions are explicit latent semantic annotations between paired structured data and texts, which support learning how to use specific phrases under the particular condition and then improve the interactivity and interpretability of the generated text compared to traditional template-based methods. More importantly, obtaining text generation trigger conditions automatically from alignment distribution could significantly reduce human editing workload compared with other commercial systems, e.g., Word-

* Contribution during the internship at Microsoft Research Asia.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 13–18
Brussels, Belgium, October 31–November 4, 2018.

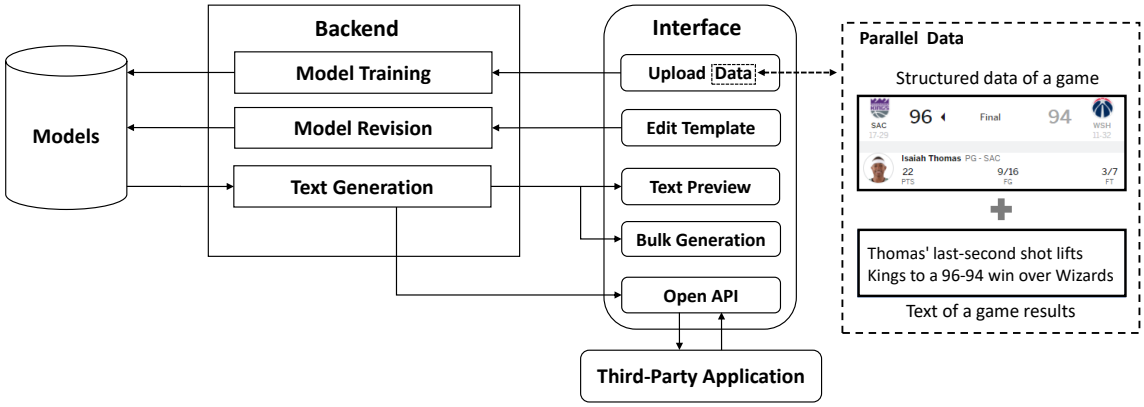


Figure 1: The simplified architecture of the Data2Text Studio.

Smith, Arria and Quill[1]. For example, although WordSmith provides functional tools to help developers create templates and generation rules, it still needs to create rules from scratch manually.

For the first challenge, we demonstrate the Data2Text Studio, a powerful platform equipped with the proposed Semi-HMMs model, to assist developers to generate texts from structured data in their own applications. Currently, this system provides several pre-trained models covering different domains: sports headline generation, resume generation, product description generation, etc. Developers can also train their own models by uploading parallel data. After model training, developers can revise the model, preview the generated texts or call the APIs to generate texts in third-party applications. All the processes are simple and friendly.

We conduct experiments on the ROTOWIRE dataset (Wiseman et al., 2017) to evaluate the performance of template extraction and overall text generation. The results show that our model achieves improvements on both tasks. The rest of this paper is organized as follows: Section 2 describes the architecture of Data2Text Studio. Section 3 proposes the main algorithm. Section 4 shows the experiment results.

2 Architecture

Fig. 1 shows the simplified architecture of the Data2Text Studio. It mainly consists of three components: 1) model training, 2) model revision and 3) text generation. For typical usages, developers can directly choose the pre-trained model to generate high-quality texts. To develop a customized text generation model: First, developers need to upload parallel data which consists of texts and corresponding structured data to train the model, and then training components will extract the templates and corresponding trigger conditions from training data automatically; secondly, developers could leverage the built-in tools to further revise the extracted templates and trigger conditions manually; finally, developers could preview the generated texts of the customized model, and the APIs are provided to generate texts in bulk or generate texts in third-party applications. In the following, we will introduce these modules in detail.

Template

[win_team_points_leader_last_name]'s
[win_team_points_leader_score] points lead
[win_team_last_name] over
[lose_team_points_leader_last_name], [lose_team_last_name],
[win_team_score]-[lose_team_score]

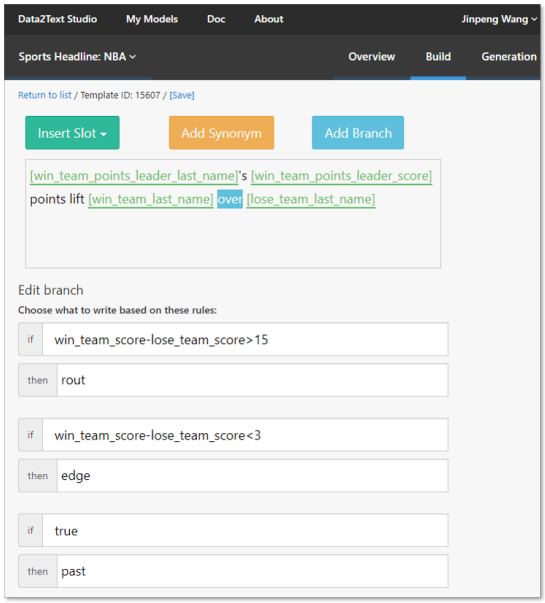
[win_team_points_leader_last_name] scores
[win_team_points_leader_score] points,
[win_team_last_name] beat
[lose_team_points_leader_first_name], [lose_team_last_name]
[win_team_score]-[lose_team_score]

Figure 2: Extracted templates of NBA headlines. Bracket indicates the slot, and words in it indicates the corresponding attribute of structured data.

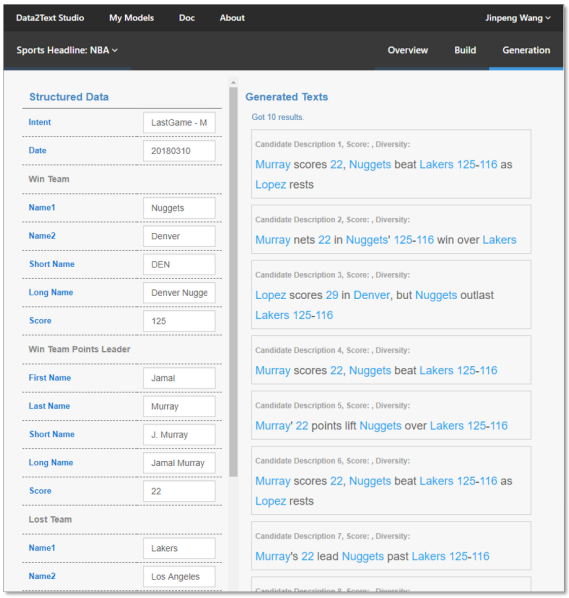
2.1 Model Training

We adopt the template-based solution for the Data2Text Studio. It can generate texts with high accuracy and fluency, which can be used in business applications directly. Several previous studies (Liang et al., 2009; Wang et al., 2017; Kondadadi et al., 2013) can be applied to extract templates from parallel data. To address the challenges introduced in Section 1, we propose a

[1] http://automatedinsights.com, http://arria.com and http://narrativescience.com



(a) Template revision. The center part shows the template with slots, and the bottom part shows the trigger conditions.



(b) Generated texts preview. Multiple headlines are generated for the same game to ensure variety.

Figure 3: Data2Text Studio Interface

Semi-HMMs model to extract templates and corresponding trigger conditions from parallel data (see Section 3.1 for the algorithm). Fig. 2 presents an example of the extracted templates from NBA Headline parallel data, which consists of the scoreboard and the corresponding news.

2.2 Model Revision

The trained model provides a better starting point for developers to avoid creating a model from scratch. If necessary, developers can revise the trained model by editing the extracted templates and their corresponding trigger conditions. Fig. 3a shows the interface of template editing. Three mechanisms are designed to manage templates and corresponding trigger conditions: 1) Data slot: the input structured data will be filled into the slot to generate texts. 2) Synonyms: it is constructed by a list of phrases, and one of them will be chosen randomly during the generation process. 3) Branch: the trigger condition to define usage scenario for the specific phrase. Our Semi-HMMs model in Section 3.1 can learn such data slots and trigger conditions automatically. Meanwhile, developers can also revise them if necessary.

2.3 Text Generation

Given the structured data, the system will generate corresponding texts with the trained model. Fig. 3b shows an example for NBA headline generation. The left-hand side shows the input structured data which contains the attributes of the game. The right-hand side shows multiple generated texts for this game to help developers check the quality of the generated texts.

2.4 API for Third-Party Applications

To use the text generation service in third-party applications, an API is created for each trained model. Once the structured data is posted through the API, the system will deliver the generated text back to third-party applications automatically. In this way, developers can leave the development work for a text generation model in the Data2Text Studio. Fig. 4 shows three application scenarios: sports headline generation, user profile generation based on LinkedIn data and car insight generation.

3 The Proposed Algorithm

In this section, we introduce the proposed algorithm for templates extraction and corresponding trigger conditions mining.

3.1 Template Extraction

A main challenge of templates extraction is the alignment between text and structured data. We adopt the model given by Liang et al. (2009), which presents a 3-tier HMMs to automatically align words to the fields of structured data. These aligned words could be strings, like brand names, or numbers copied from the data.

(a) Sports Headline (b) LinkedIn Profile (c) Car Description

Figure 4: Example applications by using the Data2Text API.

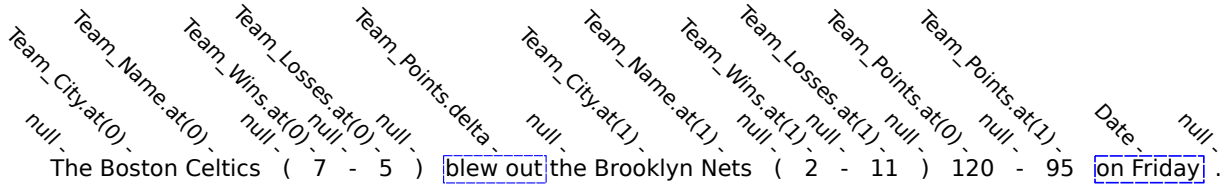


Figure 5: Example of Trigger Mechanism. Words in blue dashed boxes are particular phrases generated by our model under the specific rules. For example, *"blew out"* will be generated when *"Team_Points.delta"* overs 17 under the automatically extracted rules.

Another challenge is the lexical choice, which refers to choosing contextually-appropriate words to express non-linguistic data. For example, in a basketball game report, the author tends to use *blow out* only when the score difference is very large. Lexical choice is very subtle and differs from author to author, thus we enrich the alignment model with a Gaussian emission probability from words to numbers in the data.

The garbage collection problem is severe in the original model of Liang et al. (2009), which means that most of the words are wrongly aligned to infrequent fields that should remain unaligned (i.e, aligned to *null*). Here we incorporate the Posterior Regularization proposed by Graça et al. (2010), which could add constraints into models with latent variables while keeping the model tractable at the same time. In practice, we set a lower bound on the number of unaligned words, which could significantly alleviate the garbage collection problem.

In a nutshell, we propose a generative model, $P_s(\mathbf{w}, \pi|\mathbf{l})$, where s is the world state, namely, the structured data, $\mathbf{w}$ is the observed words, π is the segmentation of words, and $\mathbf{l}$ represents tags, which could be the fields of the structured data (e.g. `Team_Name`) or simple operations on specific fields (e.g. score difference). Let $\mathbf{c}$ be the segments of sentence $\mathbf{w}$ segmented by π. We further make a Markov assumption and factorize it into:

$$P(\mathbf{c}|\mathbf{l}) = \prod_t P(l^t|l^{t-1}) \cdot P_s(c^t|l^t), \qquad (1)$$

where c^t represents the segment at time stamp t, which is annotated with tag l^t. For different types of fields, we use different methods to model $P_s(c^t|l^t)$.

During the training process, our goal is to maximize the complete data likelihood:

$$\mathcal{L}(\theta) = \prod_{(s,\mathbf{w})\in\mathcal{D}} \sum_{\mathbf{l},\pi} P_s(\mathbf{l}, \pi, \mathbf{w}; \theta),$$

where $\mathcal{D}$ represents the whole training data. Once the model has been trained, we use Viterbi-like dynamic programming to perform the MAP inference to segment the texts and to assign the most likely tags for each span.

We derive an expectation-maximization (EM) algorithm to perform maximum likelihood esti-

mation, and introduce a soft statistical regularization to guide the model towards a better solution. Specifically, we design a special NULL tag for unaligned words, and we "encourage" it to annotate at least half of the words. For more details, please refer to Qin et al. (2018).

3.2 Trigger Mechanism

As proposed in 3.1, we use Gaussian distribution to model the probability of alignment between numerical values and phrases. Hence our model can tell us not only where the word comes from, but also the distribution of numbers it is aligned to. For example, after training, our model successfully aligns "blow out" to the score difference, and shows that the mean value of score difference is 17 when this phrase is used. With this information, we could set a "trigger" on the aligned words. Trigger is a scheme which determines under what conditions a template could be used. For example, templates with "blow out" aligned to score difference can only be used when the score difference is around 17, where *blew out* would have a higher probability than *defeated*. So we could obtain a rule like this:

```
if team_points_delta > 17:
    use(blew out)
else:
    use(defeated)
```

With such rules, our model will be able to use different words under various conditions.

Now that the templates and triggers are ready for use, for text generation, we fill the templates with structured data under corresponding applicable trigger conditions.

4 Experiments

In this section, we will report the performance of the proposed model on template extraction and on overall text generation, both evaluated on the ROTOWIRE subset of the Wiseman et al. (2017) dataset.

	Precision	Recall	F1
Liang09	0.319	0.643	0.426
Liang09+PR	0.397	0.640	0.490
Semi-HMMs	0.254	0.765	0.381
Semi-HMMs+PR	**0.504**	**0.786**	**0.614**

Table 1: Word-level tag assignment results.

4.1 Template Extraction Evaluation

We conduct an experiment and compare with Liang et al. (2009)'s system as the baseline. It is difficult to evaluate the accuracy of tag assignment for the whole dataset, since the executable tags are not annotated in the original data. We recruit three human annotators which are familiar with basketball games to label a random sample consisting of 300 sentences from the test set. The annotators were told to judge whether each word span is related to the table, and which label they are related to. Finally, we calculate the precision and recall for non-NULL tag assignments at word-level.

The results are shown at Table 1. We can observe that our initial model indeed outperforms the baseline system in recall, while posterior regularization helps a lot to avoid distraction from irrelevant information that should be tagged as NULL without sacrificing the recall performance.

4.2 Overall Text Generation Evaluation

We also test the performance of extracted templates in overall text generation, by comparing with the baseline using the same heuristics described in Section 3.2. To generate document-level texts, we first generate a sentence describing the scoreline result for every game, followed by three sentences describing other information about team performance. While maintaining that no template is repeatedly used, we then choose the template with the highest score for the top ten players sorted by their game points. We report automatic metrics including BLEU scores and those based on relation extraction as proposed by Wiseman et al. (2017): precision & number of unique relations in generation (RG), precision & recall for content selection (CS), and content ordering (CO) score. Besides these automatic metrics for various aspects in NLG, we also conduct human evaluation on information correctness (1-5 scale ratings, the higher the better). We ask four human raters who are fluent in English and familiar with basketball to rate outputs for 30 random games. Results are shown in Table 2 with Kendall's W measuring the inter annotator agreement. We can observe that templates derived from our model indeed outperform those from the baseline system.

5 Conclusion and Future Work

To summarize, Data2Text Studio is a platform for automated text generation from structured data.

Model	RG(P%)	RG(#)	CS(P%)	CS(R%)	CO	BLEU	Correctness
Liang09+PR	85.83	33.29	14.33	31.09	6.25	8.34	2.60
SemiHMM+PR	90.47	41.79	21.63	50.17	9.63	9.45	3.58
Gold-standard	91.77	12.84	100	100	100	100	4.88

Table 2: Results for text generation. (Kendall's W=0.83 from correctness raters.)

It not only provides several pre-trained models which could generate high-quality texts from data but also is very easy to train new models by uploading parallel data. In addition, this system is equipped with the proposed Semi-HMMs model which could extract templates and corresponding trigger conditions from parallel data automatically and supports learning how to use specific phrases under the particular condition. Experiment results on the ROTOWIRE dataset show that the proposed model outperforms the baseline for template extraction and text generation.

In the future, we will integrate more powerful pre-trained models into this system in terms of data domain and text fidelity. For the template extraction model, we will learn more complex grounding rules to enhance the model power.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *EMNLP*, pages 502–512.

Anja Belz. 2007. Probabilistic generation of weather forecast texts. In *HLT-NAACL*, pages 164–171.

Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *ENLG*, pages 72–81.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*, pages 128–135.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

Pablo A. Duboue and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *EMNLP*, pages 121–128.

J. V. Graça, K. Ganchev, and B. Taskar. 2010. Learning tractable word alignment models with complex constraints. *Computational Linguistics*, 36(3):481–504.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–385.

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical NLG. In *Proceedings IWCS*, pages 143–154.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING*, pages 543–551.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical NLG framework for aggregated planning and realization. In *ACL*, pages 1406–1415.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *ACL*, pages 704–710.

Rmi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *IJCNLP*, pages 91–99.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *HLT-NAACL*, pages 720–730.

Johanna D. Moore and Cécile L. Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694.

Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang, and Chin-Yew Lin. 2018. Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A statistical framework for product description generation. In *IJCNLP*, pages 187–192.

S. Wiseman, S. M. Shieber, and A. M. Rush. 2017. Challenges in data-to-document generation. In *EMNLP*, pages 2253–2263.