Constructing Statistical Data-driven Natural Language Generation Systems for Comparison Against Template-based Systems

Jarryd Dunn
University of Cape Town
dnnjar001@myuct.ac.za

ABSTRACT

Natural Language Generation is the process of generating natural language text from some set of information. The process may broadly be split into two parts text planning (what to say) and linguistic realisation (how to say it). Templates and data-driven systems are two common approaches to the problem. Using templates it is easier to produce grammatically correct utterances though the utterances often lack fluency and variation. Templates also do not generalise well to other domains. On the other hand data-driven approaches have the capacity to generate more natural and varied texts but with more errors and higher training costs. This raises the question, are data-driven approaches worth the extra cost or is it better to use a template-based approach.

Data-driven systems can be designed in many ways though most are based on neural networks. These systems may either employ an end-to-end approach or may be split into several distinct subsystems. The end-to-end approach tends to be costly and often requires multiple neural networks to be trained. Rather than using neural network another approach is to use is reinforcement learning, where policies are devices to generate the natural language utterances. These systems tend to be simpler to implement and produce less errors, though their outputs are often less varied. This implies that a successful system could be built using different approaches for different phases, for example using reinforcement learning to generate the text plan then a neural approach to realise the text plan. This provides a system that promises varied utterances, with fewer errors and lower training cost than some of the more complex data-driven approaches. Though it may not be able to generate the same variance or exhibit the same ability to reflect the style of the training data.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Natural language generation**.

KEYWORDS

NLG, Datasets, Neural Networks, Data-driven, Template-based

1 INTRODUCTION

Natural Language Generation (NLG) is the process of converting information into Natural Language utterances. This review focuses more specifically on data-to-text NLG systems, thus the meaning representations used will typically take the form of non-natural language strings. There are three main approaches to building an NLG system: modular approach, a template-based approach and a data(corpus)-driven approach[4], this review focuses on the template-based and data-driven approaches. A template-based approach makes use of one or more templates, these are either hand crafted or learnt from training data[11], data from the meaning representations is inserted into the relevant positions of the template to produce an utterance. Since there are usually only a few templates it may be possible to manually remove any errors in generated templates. This means that the utterances produced tend to have few if any semantic errors. On the other hand data-driven techniques learn from the data in order to produce natural language utterances. Thus data-driven techniques should produce more natural and varied utterances[10, 18] as they will tend to mimic the style of the data they were trained on. Though this does mean that data-driven systems will be more prone to semantic and syntactic errors since they are not explicitly trying to learn these rules but rather picking up statistical patters in the data. It is infeasible for the data-driven systems to be manually corrected once trained without adding additional algorithms to alter the utterances after the utterance has been generated. Besides not generating as varied utterances there are a few other reasons to consider data-driven systems over template-based. Template-based systems do not tend to scale as well as data-driven approaches on large open domain systems[17] where it is difficult to learn a widely applicable structure for the templates. Template models will also generally require redundant information in text in order to construct useful templates[16]. Notwithstanding, based on the systems written for the E2E NLG Challenge Puzikov et al[11] concluded that the cost of developing a complex data driven system are not worth the results and it is better to use simpler approaches.[10]

The remainder of this review seeks to preview a few data-driven systems, which use different techniques, in order to ascertain if data-driven systems may be worth the extra cost. Section 2 gives a brief outline of the basic architectures of common NLG systems. Section 3 then examines four examples of relatively successful data-to-text data-driven NLG systems. Section 4 gives an overview of some of the methods that may be used to evaluate NLG utterances and looks at the performance of the architectures analysed in section 3. Finally section 5 outlines a way forward.

2 ARCHITECTURES

The task of natural language generation may be decomposed into six subtasks[4, 13]. The organisation of these six tasks varies greatly between different architectures ranging from all six being performed by by a single module to a separate module for each task. The most common configuration makes use of two[14] or three modules[13]. The tasks are typically split among three modules (text planning, context planning and linguistic realisation) as follows:

(1) **Text Planning** takes the input for the system and produces a text plan. The text plan is an abstract structure that describes the structure and information for the output utterance. This module encapsulates the tasks of content determination and discourse planning.
Content determination involves deciding what information should included in the output text. It involves creating messages expressed in some formal structure that allows entities, concepts and the relationships between them to be distinguished.
Discourse planning involves the creation of a text plan. The text plan usually takes the shape of a tree where the leaves are messages and the internal nodes are used to determine how the leaves/messages are grouped. The text plan may also specify information about the relationship between the leaves and groups for example if one group may describe an example or elaboration of the contents of a neighbouring message. Some systems will include the direction of the relationship as an additional attribute of the relationship[6]
(2) **Context Planning** can be split into the following sub-tasks: sentence aggregation, lexicalisation and referring expression generation (REG).
Sentence Aggregation involves placing the messages from the previous stages into sentences. These sentences may be combined in order to increase the readability and naturalness of the output text. Sentence aggregation may be done based on rules, however, it can also be done by learning from the training data. By learning from the data sentences may better reflect the context of the utterances.
Lexicalisation is performed once the sentence structure has been decided in order to determine the phrases that are used to convey the concepts and relationships described by the messages. This may be done by hard coding a specific phase for each concept or relation or the phrases could be learnt from the training data.
Referring Expression Generation occurs after the phrases have been determined for the concepts and relations the words or phrases. REG involves identifying the entities involved in the text. The goal of REG is to allow the reader to distinguish each entity without providing too much redundant information, this can be achieved by using referring expressions. This task makes helps make the text appear more natural.
(3) **Linguistic Realisation**: Finally the grammar rules of the language are applied to generate a text output that conforms to the rules of the language and is syntactically (correct syntax), orthographically(conforms to the rules of the language such as capitalisation) and morphologically(correct form of the words) correct.

In a two model architecture the context planning and linguistic realisation modules are combined. The two module architecture splits up the tasks quite naturally where the first module describes what information the text should communicate while the second determines how best to convey the information. Using a two stage approach simplifies the process of generating text since it allows the first module to ignore the complexities of syntax and morphology which can be explicitly handled in the second module, however with the inclusion or a reranker to try ensure the outputs are syntactically correct a single module approach can outperform a two-stage approach [3]. The modules can be arranged such that a later model feeds into an earlier module to provide feedback and help deal with long term dependencies[13], however this architecture doesn't appear to be popular with data-driven systems.

3 DATA-DRIVEN DESIGN

In recent years there have been several different approaches to the problem of natural language generation from data using data-driven methods mostly utilising neural networks. The difference designs give rise to various strengths and weaknesses in generating natural language utterances.

3.1 TGEN

Developed by Dušek and Jurčíček (2016)[3]; TGEN v2 is a sequence-to-sequence generation approach to generate deep syntax trees or strings. A rule-based external surface realiser is then used to produce the text utterances. This does mean the the system may not generalise well to other datasets[7]. The output of the generator is then filtered using beam search and a reranker to reduce the amount of irrelevant information and linguistic errors in the utterances produced.
The inputs to the system are Dialogue Acts (DA). In the BAGEL restaurant dataset DAs consist of a predicate (which defines the purpose of the DA for example to inform or request) as well as one or more attributes and their values for example area=citycentre. Each DA is converted into a sequence of triples containing the predicate, an attribute and the value associated with the attribute. These sequences are the input the sequence-to-sequence generator. The sequence-to-sequence generator consists of two stages an encoder and a decoder. The encoder takes the input sequence and using a Long Short-Term Memory (LSTM) based recurrent neural network (RNN) produces a sequence of hidden states. The decoder, using a second LSTM-based RNN, is then initialised using the last hidden state and uses the output from the previous token at each step to produce a sequence of tokens. To increase the accuracy of the system an attention-mechanism is introduced to the decoder RNN. This consists of a feed-forward neural network which is effectively used to create an alignment model. The probability of each token is then calculated using Softmax (Softmax gives the probability for each possible outcome occurring given a finite number of possible outcomes[1]). A beam search algorithm is used to keep track of the probabilities of the n-best outputs. A reranker is then run over the n-best outputs to penalize any that are missing information or have additional information not included in the original DA. A weighted penalty, calculated based on the number of ommited or additional facts included in the utterance, is subtracted from the log-probability of the utterance. The classifier used for reranking has an architecture similar to the generator using a RNN as an encoder but only a single logistic layer as a decoder.
This system improves on the original TGEN system, TGEN v1, created by Dušek and Jurčíček (2015)[2] by making use of the LSTM-based RNNs and beam search rather than perception scorer and A*

searching. A perception scorer is in less powerful than an LSTM-based RNN since it essential consists of a single weighted edge while an RNN has multiple nodes where the output from a node may be fed back to previous nodes essentially giving the network memory.
The encoder-decoder paradigm is widely used since it enables a variable-length sequence of input tokens to be mapped to a variable-length sequence of output tokens[4]. LSTM-based RNNs are used for the encoder and decoder in TGEN v2 since they are supposed to better capture long distance dependences between tokens compared to a normal RNN[5]. A disadvantage of the is approach is the computational cost of training three separate RNNs.

3.2 Joint RNN and CNN System (RNNLG)[18]

RNNLG is an end-to-end system meaning that the modules of text planning and linguistic realisation(the process of forming natural language utterances from the text plan) are not separated. As in TGEN this system uses an over-generate and rerank approach, however, unlike TGEN it generates natural language utterances without the need of an external surface realiser. A generator is used to create multiple possible options before using a reranker to select the most coherent utterances.
The RNNML used for the generator is trained over a delexicalised corpus. In a delexicalised corpus the values for a dialogue act attribute are replaced with a token type. Thus the machine learning algorithm can learn how to deal with a entity in general (e.g restaurant names) rather than a specific instance of a entity. This approach does run into difficulties where there is no single values to tokenise for example the attribute dogs_allowed can be represented in may different ways such as "pet friendly" or "dogs allowed". It is far more complex for the system to recognise that the phrase "pet friendly" means that dogs_allowed=true then to swap a single word or phrase with a token. This tends to lead to mistakes in the output utterances. It is also not always desirable to delexicalise an attribute, for instance if it appears several times in the corpus (the same area may be mentioned several times) then by delexicising the attribute some of the more context specific language relating to the attribute will not be learnt.
A recurrent neural net language model (RNNML) is used for the generator. The generator operates on a 1-hot vector encoding of a DA, representing which attributes are included, and a delexicalised utterance. A 1-hot vector representing a token forms the input at each time step. The vector is used as a gate to try and reduce the probability of the same slot featuring multiple times in an utterance. The output from the RNNML is a probability distribution, calculated using Softmax, for the next token based on the hidden states. A delexicalised output text is found by choosing the most likely output tokens until some criteria is fulfilled such as an end of sentence token being reached. By replacing the delexicalised tokens in the sequence with the corresponding values from the DA an output utterance can be produced; however before the token sequences can be lexicalised they are reranked to try improve the quality of the utterance.
The reranker consists of two modules; a convoluted neural network (CNN) sentence model and a backward RNNML which is used to rerank the outputs.
The sentence model produces a matrix based on the token contained in a given candidate utterance. This matrix is then passed to a feed-forward neural network which classifies the utterance type as well as if it contains values for the required attributes, to try ensure all the facts from the input are represented. The output of the sentence model is a feature map for the candidate utterance. Including the CNN should decrease the number of missed and additional facts compared to the input DA.
The feature map is given as input to the backward reranker. The backward reranker is a second RNNML. The backward reranker computes the log-probabilities for each of the candidate utterances produced by the forward reranker. The quality of the outputs was further increased by adding an error term to penalise any slots from the input DA that were not included in the output or the inclusion of additional attributes that were not in the input DA.
The two RNNMLs are separated by the sentence model CNN to try and replicate the behaviour of an bidirectional RNN, since bidirectional RNNs have been shown to outperform RNNs at tasks such as handwriting analysis[18]. The authors found it was not possible to directly apply a bidirectional-RNN since it is sequential in time.
Many of the data-driven systems use some form of neural network in order to generate text (such as the previous two). The use of LSTM-based recurrent neural networks is particularly popular ince there is evidence that LSTM cells are good at dealing with long term dependencies between n-grams[15]. These dependences are common in natural language so LSTM-cells should enable a system to accurately learn more complex behaviour. An alternate way to try capture these dependences is to use gated recursive units (GRU) [11, 16]. GRUs outperform LSTMs for most tasks though language modelling appears to be an exception[5]. The performance of a system using an LSTM-based RNN can be improved significantly by using a deep LSTM-based RNN[15]. It may also be increase by adding a large forget bias[5]. Other approaches include using reinforcement learning to learn the most appropriate outcomes based on some reward/loss function (such as the next system, LOLS).

3.3 Locally Optimal Learning to Search (LOLS)[6]

LOLS like the TGEN systems features a sentence planner and surface realisation module. Also similar to the RNNLG system a delexicalised corpus is used to train the imitation learning frame work. LOLS takes a different approach to the previous two architectures essentially viewing NLG as a classification problem by grouping words that are related to attributes from the input DA then using them to express the attribute. The system first predicts the content of the utterance by predicting a sequence of attributes one by one. Each attribute is selected based on its predecessors. Once a sequence of attributes has been found words are selected, from a dictionary corresponding to the the particular attribute, to express the attribute. The dictionary for a particular attribute is learnt from the text based on words that appear in close proximity to the attribute.
A policy must be learnt in order to determine the sequence of attributes/tokens that will make up the content prediction. Similarly

a word prediction policy must be learnt to realise the content tokens into a natural language utterance. The policies are trained separately but are learnt in the same way, using a set of training examples and a loss function. These policies are used to generate the next token given the current token and all previous tokens that have been generated. The loss function is used to compare the sentences produced by the policy and the reference utterances given in the training set. The policy is then updated based on the loss function. One of the natural language evaluation metrics, such as BLEU, is typically used as the loss function. Using an evaluation metric means that the utterances that the trained policy produces should score highly when evaluated with the metrics it was trained on and avoid certain mistakes such as repeated words or information not in the reference text; however, this does not guarantee it will score highly when judged by people[9]. The learnt policy is trained over multiple iterations, in each iteration the sequence of content tokens is generated by either an expert reference policy or the learnt policy from the previous example. The reference policy is derived from the reference sentence, so is only available for the training data, while the learnt policy should be able to generalise and so produce token sequences for unseen DAs. To ensure an optimal learnt policy at each iteration the algorithm must decide if it should exploit a known token or set of tokens or try explore different tokens to try and increase the loss function (produce better utterances). The exploration is done by using the reference policy. The reference policy is chosen over the learnt policy from the previous iteration with probability $p = (1 - \beta)^i$ where β is the learning rate and i is the number of iterations. This allows the learnt policy to be chosen more frequently as it gets better at generalising. To speed up the training and avoid mistakes a sequence correction mechanism is incorporated to correct sub-optimal choices. To avoid over fitting the sequence correction mechanism looks at the sub-optimal token and the next E tokens generate before deciding if it would reduce the cost function to rather use the reference policy to generate the next E tokens. This helps solve a common problem with NLG systems where errors tend to propagate as tokens are generated one at a time based on previous tokens.[12]

In order to generate a natural language utterance a delexicalised DA is used with the content classifier to generate a set of tokens. This set of tokens is then used with the word classifier to generate an utterance. Finally the utterance will be delexicalised by inserting explicit values from the DA, this is essentially the reverse of the process carried out to delexicalise the training corpus.

3.4 Modular Approach[8]

Again this system is split into two modules; text planning and surface realisation. Unlike RNNLG which is an end-to-end system, this system separates the tasks of learning text planning and surface realisation. By separating these tasks the two subsystems each have a simpler set of rules to learn, thus should be able to produce better results.

The text planning phase takes a similar approach to LOLS, where a plan is learnt it decides what facts to include in each sentence and the order in which the sentences should appear. Additionally within each sentence the text plan determines the ordering of entities within each fact and the structures between facts. To achieve this the text plan consists of a sequence of sentence plans where each sentence plan is a tree. The trees give the order in which the facts should be realised. The text model assumes that each entity only appears once per sentence thus it is not clear how well the system would work on training sets that include multiple occurrences of an entity in a sentence.

The WebNLG corpus[1] is used to train the text plan. The DAs in WebNLG consist of a set of triplets (s, r, o) where s is the subject, o the object and r the relationship between the subject and the object. Since the WebNLG corpus contains small sets of DAs the text plans may be generated by constructing an undirected graph between the entities then using a depth first search (DFS) to generate a text plan. By starting the DFS at each node and visiting the descendants in a different order all text plans may be generated. The edges of the graph take the form of a triple (h, l, m) where h and m are entities, corresponding to the subject and object, while l consists of a relationship, r, and it's direction. Representing l as a relationship and direction rather than just a relationship allows the subject and object to be swapped, this means that text can be generated in either a passive or active voice. For larger inputs generating all possible text plans may become impractical though some heuristic could potentially be used to reduce the number of text plans generated. The structure of the edges makes it easy to identify sentences that may be combined since either h or m will be common in both triples. A text plan is considered to be consistent with the reference (this makes it more likely to produce a valid utterance containing the required facts) if it uses the same sentence splits and has the entities in the same order as the reference text for each sentence plan. The system is not strict on matching entities since many entities may be realised in several different ways making it difficult to determine if an entity has a match, this is particularly difficult when context and external knowledge is required to know two entities are equivalent (e.g nicknames). Another issue is that while matching the order of entities reduces the chance of mistakes being made in the sentence, it may also reduce the variation shown in generated texts. A product-of-experts approach is used for plan generation, this involves evaluation the plans using several experts then multiplying the results together and normalising the result to get a final score for the text plan. The experts used are: Relational direction, calculated as the probability of the direction of the relationship in the plan; global direction, probability of observing n reversed relationships (from the MR) in x triplets; splitting sentences, the probability of a specific distribution of the number of facts amongst sentences.Using this approach not only allows different plans to be compared but a threshold can be found for generating good quality utterances. By selecting a random plan above the threshold more variance might be shown in the system while still ensuring good quality utterances are produced.

Splitting the text plan into sentence plans makes the realisation process easier since each sentence plan can be realised individually. The downside to this approach is that it cannot include some structures such as referring expressions (potentially making the utterances less natural). In order to realise the sentence plans they are first linearised by performing a pre-order tree traversal. A neural machine translation algorithm is then used to determine the words

[1]Documents for the corpus can be found here http://webnlg.loria.fr/pages/docs.html

to realise the sentence plan. Similarly to the previous systems a delexicalised text plan is input to the NMT and the NMT learns over delexicalised data. The tokens in the output of the NMT are then replaced based on the original input graph.
The utterances produces in general contained less additional facts and omit less facts than a strong neural based system it was tested against. The fluency of the system was measured by performing pair-wise evaluation between two utterances using human judgement to select which utterance better described the reference text. As expected judges found the reference text to be more fluent, however, the system was as fluent as the neural system and out performed the grammar-based system it was tested against.

4 SYSTEM PERFORMANCE AND COMPARISONS

The performance of a system may be measured by running evaluation metrics on their output utterances, however, a fairer comparison between systems may be achieved by running both systems on the same data set and comparing the evaluation scores achieved.

4.1 System Evaluation Metrics

There are several automated metrics that may be used to evaluate natural language utterances. These metrics are useful since they provide a relatively fast and inexpensive way to judge generated texts. Metrics may generally be classed as word-based or grammar-based. Word-based metrics use several features to try and evaluate an utterance:

- Word overlap and string distance: This is based on n-grams common to both the generated and reference text. For example BLEU which measures the accuracy with which the utterance captures the information of the ground-truth reference or ROUGE which measures the recall.[6] The disadvantage of using BLEU is that it considers the length of texts thus shorter reference texts are likely to receive lower scores[4]. While string distance is a measure of the number of insertions, deletions, substitutions and transpositions that would need to be applied to convert the generated text to the reference text. TER is a string distance metric.[4]
- Semantic similarity, based on the similarity of word distribution and semantics analysis.[9].

In general the higher the score the more similar the generated text is to the human reference, TER is an exception where the scale is reversed[9]. Though these scores may not always to representative for instance most word-based metrics struggle with recognising synonyms.[4]
An alternative to word-based metrics are grammar based metrics. These use grammars to evaluate the grammatical correctness and readability of text. An advantage of grammar-based metrics over word-based metrics is that no reference text is required, this means that errors in evaluation cannot occur due to faulty reference texts (this is a risk particularly when reference texts are crowd sourced). Grammar-based metrics may be used to measure readability and the grammatical correctness of a text. Readability refers to how easy a text is to understand. Metrics such as Flesch Reading Ease score maybe used to measure readability based on the number of characters, words and symbols present in a sentence. The Stanford parser may be used to calculate a parsing score to measure how grammatically correct a sentence is.
The *ERR* value, which is used in several systems[18][6][17], gives a good idea of how well the utterance generated reflects how well the generated text captures the facts from the input MR. This is very useful since the purpose of the NLG system is to represent the facts in the data as text, thus it can help prove how reliable the score of a more complex evaluation metrics is (by leaving out information an utterance doesn't need to be as complex so may score higher on some evaluations)
Evaluation metrics do not negate the need for human judges since there is little correlation between the score produced by evaluation metrics and human evaluation of utterances[9]. Thus human judges provide valuable insight into the performance of a system, especially since the systems are designed to produce text for humans. Human evaluations of systems maybe done as pair-wise comparisons, essentially an AB test, displaying two generated utterances for a DA and asking human judges which they prefer. This could be done in addition to a quantitative judging of the texts based on some criteria most likely informativeness, fluency and naturalness. The AB test could reveal how humans perceive the text without analysing it too closely, it is possible that by making the judges pay closer attention to the text, in order to try and score it, the judges would be changing their perceptions of the text. AB testing would, however, not provide much insight into why people preferred one text over another while qualitative testing could. There is evidence that a AB testing approach to evaluation is more sensitive and has less variance[4]. Though qualitative evaluation may be improved by using a visual scale such as a slider for grading rather than getting the judges to assign numerical scores[4]. The reference texts are required to be shown along side the generated text so the judges can determine how informative the texts are.

4.2 System Comparisons

Lampouras and Vlachos[6] compared the performance of their NLG system, LOLS, against RNNLG(WEN)[18] over the SF dataset which contains meaning representations(MRs) for hotels and restaurants in San Francisco. The results are shown in table 1. RNNLG consistently score higher in both the BLEU and ROUGE metrics, however, if the DAs where lexicalised then there was some overlap between the training and test sets (Resturant1 is in Newlands and Resturant2 is in Rondebosch both would become <Restaurant-name> is in <Area>). Since both systems are trained on using a delexicalised corpus they would have seen some of the DAs used for testing in training. Once the test set was reduced to only those where there was no overlap between the test and training sets for delexicalised DAs the difference in the metrics score narrowed significantly. Though this test set was small (11 in the hotel and 13 in the restaurant dataset) so limited conclusions can be drawn from them.
Lampouras and Vlachos also compared their system against TGEN[2] (table 2) where it was found that the TGEN v1 system scored higher using the BLEU and ROUGE metrics but lower using NIST metric and it had a higher *ERR* ($ERR = \frac{\text{attributes present+attriutes excluded}}{\text{attributes present in DA}}$) score than LOLS. A lower ERR score indicates that LOLS is better at including the facts from the DA in the output utterance. When

SF Restaurant									
	FULL TEST (1040 MRS)			UNIQUE (158 MRS)			NON-OVERLAPPING (13 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
RNNLG	74.50	77.75	2.54	52.97	43.52	6.29	27.04	20.44	10.38
LOLS	66.01	64.56	0.15	49.44	38.52	0.58	28.21	21.47	0.00

SF Hotel									
	FULL TEST (1076 MRS)			UNIQUE (96 MRS)			NON-OVERLAPPING (11 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
RNNLG	86.54	84.36	0.88	66.37	56.19	3.99	37.24	27.27	6.82
LOLS	80.00	76.88	0.25	68.65	68.37	0.52	33.31	27.01	3.63

Table 1: Results of the comparison of LOLS and RNNLG[6]

BAGEL				
	BLEU	ROUGE	NIST	ERR(%)
TGEN	56.71	48.08	5.441	24.16
LOLS	54.22	47.39	5.547	16.66

Table 2: Results of comparison between LOLS and TGEN v1[6]

	BAGEL		SF Resturant		SF Hotel	
	TGEN v1	LOLS	RNNLG	LOLS	RNNLG	LOLS
Fluency	5.15	4.79	4.49	4.23	4.41	4.68
Informativeness	4.53	5.24	5.29	5.36	5.36	5.19

Table 3: Results of comparison between TGEN v1, LOLS and RNNLG[6]

Lampouras and Vlachos used humans to evaluate the three systems (table 3), based on informativeness and fluency, it was found that TGEN v1 and RNNLG where in general more fluent but less informative than LOLS. Though compared to RNNLG using the SF hotel dataset the reverse was true and LOLS was deemed to be more fluent but less informative.

5 FINDINGS

Based on the current research data-driven NLG systems show great capacity for generating fluent, natural and varied utterances, however, they do tend to produce more errors than template-based systems. In general the systems based on neural networks tend to produce more varied and natural texts than than other systems, however they also tend to produce more linguistic errors than the data-driven systems that rely on non-neural mechanisms based NLG systems. This suggests that a solution similar to Moryossef et al.(2019) could provide a promising way forward since their system was able to generate utterances that are informative, fluent (table 3) and contain few errors(table 1 and table 2).

A potential problem for constructing a data-driven methods is noisy learning data[9], thus the corpus used for training should be carefully selected especially since it seems that noisy data may decrease the performance of a data-driven system more than a template-based system.

Word-based metrics can serve as valuable testing mechanisms since they are quick and easy to use. There is, however, a need for human judges since these evaluation metric are weakly correlated to human judge's evaluations [9]. In order to comprehensibly compare systems it should be necessary to use some form of human evaluation.

REFERENCES

[1] [n. d.]. Multi-Class Neural Networks: Softmax | Machine Learning Crash Course | Google Developers. https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

[2] Ondřej Dušek and Filip Jurcicek. 2015. Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 451–461.

[3] Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491* (2016).

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

[5] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*. 2342–2350.

[6] Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1101–1112.

[7] Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1101.

[8] Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. *CoRR* abs/1904.03396 (2019).

[9] Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why We Need New Evaluation Metrics for NLG. *CoRR* abs/1707.06875 (2017). arXiv:1707.06875 http://arxiv.org/abs/1707.06875

[10] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254* (2017).

[11] Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*. 463–471.

[12] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* (2015).

[13] Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3, 1 (1997), 57–87.

[14] Koenraad Smedt, Helmut Horacek, and Michael Zock. 1996. Architectures for natural language generation: Problems and perspectives. In *Trends in Natural Language Generation An Artificial Intelligence Perspective*.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[16] Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a Knowledge Base. In *Proceedings of the 11th International Conference on Natural Language Generation*. 10–21.

[17] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *CoRR* abs/1508.01745 (2015). arXiv:1508.01745 http://arxiv.org/abs/1508.01745

[18] Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755* (2015).