Neural Natural Language Generation with Unstructured Contextual Information

Author: Harritxu Gete

Advisors: Thierry Etchegoyhen and Oier Lopez De Lacalle

hap/lap

Hizkuntzaren Azterketa eta Prozesamendua
Language Analysis and Processing

Final Thesis

September 2018

Departments: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineer.

Laburpena

Lan honetan, hizkuntza naturalaren sorrera automatikoan informazio ez-egituratuaren esplotazioak izan dezakeen eragina aztertzen da. Bere helburu nagusia, sistema batek aurrez ikusi gabeko informazioa erabiliz testu koherentea sortzeko duen gaitasuna ebaluatzea da. Corpus berri bat ere aurkezten da, zeregin honetarako bereziki prestatutako Amazon Review corpusaren aldaera bat, produktuen deskribapenak input gisa erabiliz, erabiltzaileen iritziak automatikoki sortzeko erabiltzen dena. Hainbat deep learning ereduk eginkizun honetan lortzen dituzten emaitzak konparatzen dira eta informazio ez egituratua ustiatzeko gaitasun maila ezberdina dutela erakusten da.

Abstract

In this work, we present a novel task for automatic natural language generation, based on the exploitation of unstructured contextual information. The main aim of the task is to enable the evaluation of a system's capability to generate coherent text based on previously unseen and unstructured information. A new corpus was prepared specifically for the task, based on the Amazon Review corpus with product descriptions used as input for the generation of user reviews. Different deep learning generation models were implemented and compared under the proposed task, with significant differences in terms of their ability to exploit unstructured contextual information.

Contents

List of Figures

List of Tables

1 Introduction

One of the biggest challenges in the age of Big Data is the interpretation of the generated data stream. Textual descriptions of the data would help to understand and exploit the available information, but with currently large and growing data volumes, the generation of text from data cannot be performed manually. In this type of scenarios, Natural Language Generation (NLG) technology becomes necessary.

NLG is defined as a subfield of Natural Language Processing (NLP) and Artificial Intelligence (AI) that consists in automatically generating understandable coherent text from raw data or abstract meaning representations. Through the use of NLG technology, data can be expressed in a more practical way for human understanding and use. The input data can be textual or not, and text-to-text and data-to-text generation are thus both instances of NLG. Some examples of NLG applications are automatic generation of meteorological reports and textual explanations of numerical graphs. Additionally, NLG can be used to produce texts that would otherwise not be written, with NLG being used for instance to automatically generate articles for secondary sport games or news that are not covered by journalists. In 2014 for example, the Los Angeles Times became the first media outlet to communicate a minor earthquake that took place in California, reported within 3 minutes of the occurrence using a previously written algorithm and the automatically registered earthquake data.

An NLG system could also produce different texts from the same input data, personalising the texts depending on the expected reader. For instance, in clinical contexts, it seems appropriate to explain medical data with more technical details if the expected reader is medical staff, and with more explanatory words if it is written for the patient. Whereas it would be remarkably time-consuming for a person to create all these adapted texts, this is feasible via an adequate NLG system.

Although the field of NLG originated several decades ago, for historical reasons, and because of the complexity of the task itself, it has remained in the background within NLP. With the recent success of deep learning in NLP (automatic translation, speech recognition, etc.), however, key NLG tasks have been given a new impulse, with the hope that this new paradigm of artificial intelligence will be able to resolve them. This new approach to NLG has been successful and has become an important component in multiple applications including dialogue systems, text summarisation, image captioning or video description, with considerable commercial interest in different NLG applications such as summarisation of financial and business data. It is therefore of importance to carry out research that explores the capabilities and limitations of deep learning for NLG.

The objective of this work is to develop and adapt a methodology and training framework for NLG using deep learning techniques. More specifically, the goal of this study will centre on the generation of book reviews using input information composed of book descriptions and user ratings. Automatic review generation has gained interest in the last years, with the rapid growth of e-commerce that has produced millions of human-written reviews of multiple products that enable the training of deep learning systems on this task.

Despite the large volumes of available information, the input information related to

products is very restricted in most recent studies. The most common approach involves the use of a numerical value that identifies the product, but since this number does not provide any information about the product, this procedure prevents the system from generalising to new products that do not appear during training. Since using some predefined characteristics of the product has already been explored in the literature, we aim to explore the capabilities of using unstructured data as input to enable the evaluation of a system's capability to generate text, based on previously unseen and unstructured information.

In this work, book descriptions are selected as unstructured input and, to our knowledge, this is the first work that explores the use of this type of data to generate reviews without the aforementioned dependency on previously observed training instances. We also aim to analyse ways to resolve some of the current limitations of using deep learning for NLG, such as out-of-vocabulary words or the balance between generated text coherence and output variability. Additionally, we also perform an exhaustive analysis of different evaluation methods in order to highlight the challenges in NLG evaluation, with an emphasis on several automatic metrics and the relationships between them. Finally, we evaluate different deep learning NLG architectures on the proposed task based on unstructured input.

To complete our goals, we prepare and describe a new corpus, which has been structured and prepared for the analysis of the performance of NLG systems on the specific task of generalising to new products. The dataset is composed of online book reviews and product overlaps between the different sets in the data partition are avoided, providing robust training and evaluation data for the task. The dataset notably includes different test sets split in terms of content complexity via perplexity measures.

Various well-known NLG architectures are selected with the aim to exploit the information provided by the descriptions and trained on this dataset, via the inclusion of state-of-the-art mechanisms in the NLG domain. Additionally, we develop and evaluate an architectural variant based on Latent Dirichlet Allocation which has not been previously explored for the automatic review generation task. The impact of each architectural variant on the task is measured by means of different metrics on various test sets to support a fine-grained analysis of the results, thus testing their respective generation capabilities on unseen products via the unstructured contextual information provided by the descriptions.

This work is structured as follows: Section 2 describes the theoretical framework and the state of the art in NLG, with an emphasis on automatic review generation; in Section 3, we present the current evaluation methods for NLG systems; Section 4 describes the different types of input data for NLG with their respective pros and cons, together with a detailed description of the corpus that has been used and how it has been prepared; Section 5 describes the architectures used in the experimentation, which is presented in Section 6; Section 7 presents the results, their evaluation and an error analysis; finally, Section 8 summarises the conclusions from this study and discusses future work.

2 Theoretical Framework

This section provides the basic background on deep learning needed to understand the methodology and experiments in this work. It also describes the evolution of NLG systems, from the first traditional approaches to more recent neural methods based on deep learning. Additionally, the state of the art in NLG and automatic review generation is addressed.

2.1 Deep Learning

Deep Learning (DL) is a subset of machine learning techniques, and thus an aspect of artificial intelligence, that composes multiple non-linear transformations with the goal of yielding more abstract and useful representations of data (Bengio et al., 2013), such as word embeddings vectors used in NLP to represent words instead of a unique dimension via one-hot vectors. The term *deep* refers to the composition of multiple levels, each of them learning to transform the input into a more abstract representation. A good data representation assigns greater weight to different aspects of the input depending on the task. For image classification, for example, it has to highlight those attributes that are helpful to determine the object in the picture and overlook irrelevant ones, such as the background.

Conventional machine-learning techniques require feature engineering to obtain good feature combinations that can introduce more dimensions to the input, transforming it into a space where the data-points are closer to being linearly separable (Goldberg, 2016). Therefore, it is essential for the proper functioning of this methods to come up with an effective combination of features within a very large space of possibilities. DL, on the contrary, allows the automatic discovering of good representations of the data using a general learning procedure that is capable of learning complex functions without the need for hard feature engineering.

DL models are based on artificial neural networks, a biologically-inspired programming paradigm which enables a computer to learn from observational data (Goodfellow et al., 2016). As they constitute part of machine learning, they can be trained using supervised or unsupervised learning processes, depending on whether the training data is labelled or not. In supervised methods, the algorithm learns a function to map the input into the output, while in unsupervised ones, it learns the implicit structure of the input data (e.g. clustering). Other techniques such as semi-supervised learning or reinforcement learning are out of the scope of this work, which is mainly focused on supervised learning since it is the approach used in our experiments.

2.1.1 Artificial Neural Networks

According to Schmidhuber (2015), an artificial neural network is a learning model that consists of many connected units called neurons, each taking one or various inputs and producing an output. Both the inputs and the outputs take real values, and the output is computed by means of a non-linear function which is called the activation function.

The neurons are arranged in layers indicating the course of information: the first and the last layers are the input and the output of the network respectively, and the layers that may be between them are called hidden layers. Neural networks containing several hidden layers are called deep neural networks and they can approximate complex mathematical functions, provided they contain a sufficient number of neurons.

Artificial neural networks have evolved over the years, from the perceptron, a precursor to the current neural networks Rosenblatt (1958), to the first practical multilayer networks of Ivakhnenko and Lapa (1965). After years of being relatively left aside, the active interest that neural networks have recently generated has led to the current existence of many types of neural networks.

Artificial neural networks are classified as feedforward or recurrent networks, depending on their internal connections and on how the information flows within them. We describe each one in turn below.

Feedforward Neural Networks

Feedforward neural networks (FNN) map a fixed-size input to a fixed-size output. For that aim, each unit computes a weighted sum of the outputs in the previous layer and applies an activation function to the result.

Although this is the simplest type of neural network, with the composition of enough hidden layers a FNN may be able to distort the input and convert non-linearly separable problems into higher dimensional linearly separable ones.

FNNs are very effective for classification tasks, where the number of neurons in the input is the number of features, and each neuron in the output layer corresponds to a different class. In this case, the activation function of the output layer is typically a softmax function for multi-class classification and a sigmoid function for binary classification. Both functions return a value between 0 and 1 that can be understood as the probability of the input to belong to the corresponding category.

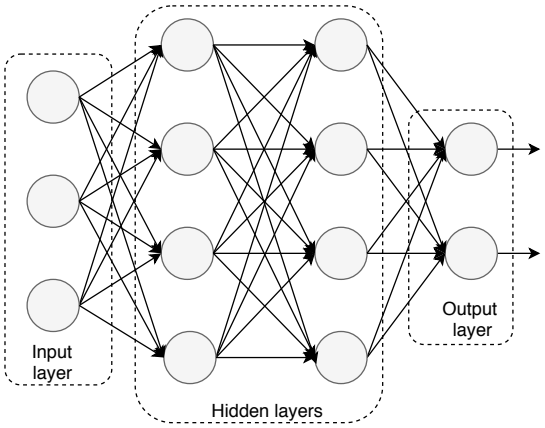

Figure 1: MLP with two hidden layers

A common FNN is the **multilayer perceptron (MLP)** (Figure 1), a type of FFN with at least one hidden layer and fully connected layers, i.e. where each neuron is connected to all of the neurons in the subsequent layer. They have achieved better results than linear models on various tasks including syntactic parsing and sentiment classification (Goldberg, 2016), and they are also referred to as "vanilla" neural networks when they have a single hidden layer.

Equation 1 describes the calculations made by a hidden unit with n inputs, where w_i are the weights of the inputs, b an optional bias term associated with the neuron and φ the activation function.

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$

The computation of the whole layer can be expressed together using matrix notation as in the following equation:

$$\mathbf{y} = \varphi\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{2}$$

In what follows, we will use this conventional matrix notation, where bold capital letters represent matrices and vectors are represented by lower case bold letters. More specifically, $\mathbf{W}$ will refer to the matrix containing the weights of the inputs and $\mathbf{b}$ will refer to the vector containing the biases. All vectors and matrices can take real values.

Recurrent Neural Networks

A limitation of FNNs is that the input is assumed to have a fixed dimension specified before training. However, in NLP, we often need to handle sequences of indeterminate sizes like texts. Besides, FNNs encode sequences as fixed size vectors, sacrificing most of their structural information. Recurrent neural networks (RNN) are specifically designed to preserve the temporal information of sequences and deal with inputs of arbitrary lengths.

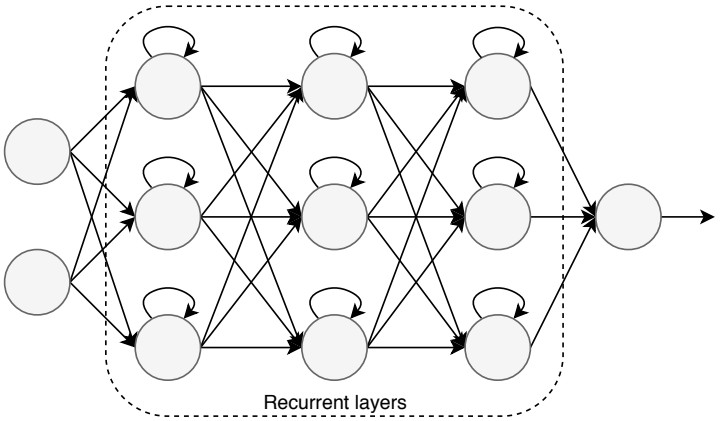

Figure 2: RNN with three hidden layers

The idea behind RNNs was introduced by Elman (1990) and consists in running the network several times, feeding it with an element of the input sequence at each time step. Recurrent architectures can be viewed as extremely deep neural networks with weight sharing across time, because in addition to passing the output of the hidden units to the next layer, they also provide feedback to themselves at each time step, as depicted in Figure 2.

RNNs have produced remarkably strong results for multiple NLP tasks including language modelling, machine translation, dependency parsing, sentiment analysis, noisy text normalisation and response generation (Goldberg, 2016).

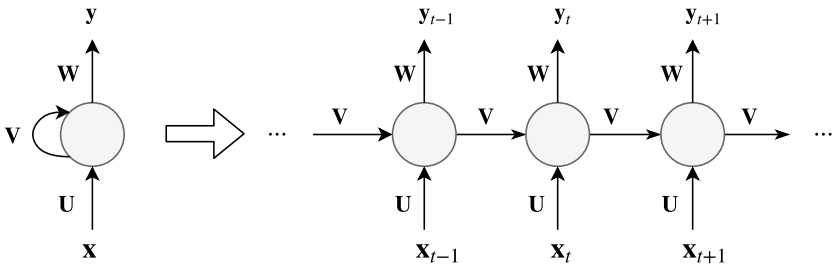

Figure 3: A hidden unit of a RNN, unrolled

There are various types of RNN neurons, but perhaps one of the most famous is the **long short term memory unit (LSTM)** (Hochreiter and Schmidhuber 1997). It has been essential to the success or RNNs, as it produces better results than the standard version in many tasks, mainly because they are capable of learning long-term dependencies that are difficult for standard RNNs.

The core idea behind LSTMs is that there is a cell state that stores all the information accumulated by the unit, and the neuron has the ability to learn the information that needs to be removed or maintained in the cell state. The flow of information is regulated by three structures called gates: the forget gate decides what information must be removed from the cell state, the input gate decides what new information is going to be stored and, finally, the output gate selectively decides which part of the cell state to output as the new hidden state.

For each element in the input sequence, each neuron computes the following equations, where: $\mathbf{h}_t$, $\mathbf{c}_t$ and $\mathbf{x}_t$ are the hidden state, cell state and input at time t; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget, and output gates; and σ is the sigmoid function:[1]

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \tag{3}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \tag{4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{(t-1)} + \mathbf{b}_o) \tag{5}$$

[1]The sigmoid function is defined as: $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{(t-1)} + \mathbf{i}_t \tanh(\mathbf{W}_{ig}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{(t-1)} + \mathbf{b}_g) \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t) \quad (7)$$

2.1.2 Supervised learning, gradient descent and backpropagation

The objective of DL is to accurately assign weights to make the neural network exhibit the desired behaviour (Schmidhuber, 2015), and supervised learning is the most common form of achieving this goal. During supervised training, outputs produced by the network are compared to ground truth results, and the generated error is measured using a function which is called the objective function (also known as loss function or cost function). The network then adjusts its internal parameters (weights and biases) to reduce the error, and the process is repeated until the errors produced by the network in terms of the objective function do not decrease anymore.

The parameters of the network can be properly adjusted and thus, the cost function minimised, via **gradient descent**. This procedure consists in modifying the weights according to the gradient of the cost function, which indicates, for each weight, by what amount the error would increase (or decrease) if the weights were increased by an infinitesimal amount (LeCun et al., 2015). Each weight is adjusted in the direction opposite to the gradient, according to the following equation, where θ is the parameter we want to update, J the cost function and α a real-valued number.

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta) \quad (8)$$

To intuitively understand gradient descent, the cost function can be seen as an irregular landscape in the high-dimensional space of weights. The gradient vector indicates the steepest slope, and thus, the direction that conducts us closer to the minimum we want to achieve. The size of the step is controlled by a hyper-parameter called learning rate, which is represented as α in equation 8. Figure 4 illustrates gradient descent when the number of parameters to update is one or two, keeping in mind that neural networks used in real applications typically have millions of parameters.

A limitation of this method is that computing the gradient over the entire training set can make the convergence very slow. Stochastic gradient descent (SGD) makes use of a subset of the training instances to calculate an estimation of the gradient, which is much faster. On the other hand, if we perform SGD to compute the gradient and perform an update for each training example, the high variance of the updates makes convergence difficult.

Performing an update for every n instances (mini-batch gradient descent) is the most used approach by DL practitioners, because it reduces the variance of the updates, leading to a more stable convergence. However, there are still some challenges that need to be addressed, such as the possibility of getting stuck in a local minimum when the error surface is non-convex, or the selection of the learning rate. While an excessively small learning rate can lead to slow convergence, if it is too large gradient descent can overstep

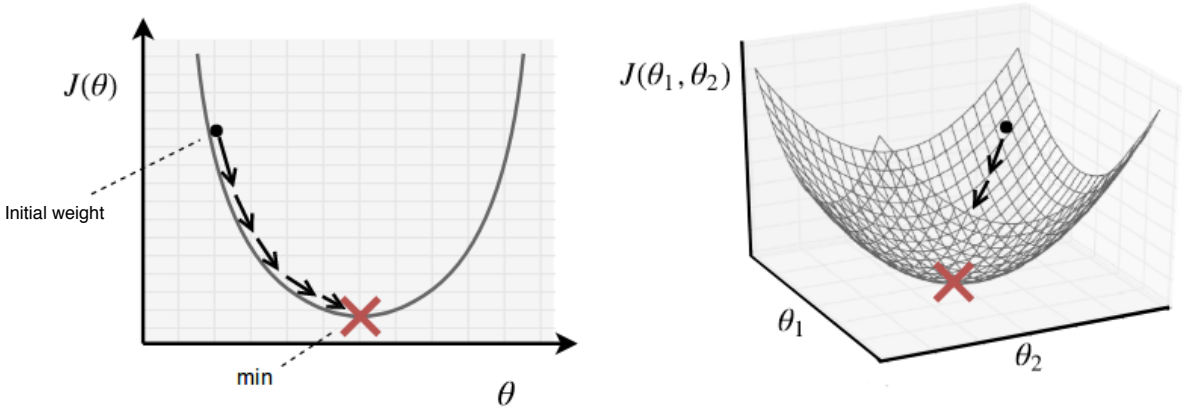

Figure 4: Schema of gradient descent with one and two dimensions

the minimum and even diverge. However, it has been shown that decreasing the learning rate during training helps SGD to converge to a minimum (Krogh and Hertz, 1992).

In order to perform gradient descent, we need to calculate the gradients of the objective function with respect to all the parameters in the network. **Backpropagation** (Rumelhart et al., 1986) is an efficient method to compute them, based on the application of the chain rule for derivatives. Applying it repeatedly, gradients can be propagated through all layers, all the way to the input. The computation of the network is therefore composed of a forward pass (computing the output using the input features) followed by a backward pass (calculating the gradient via backpropagation and updating parameters).

Backpropagation can be applied to train RNNs if we consider the outputs of a unit at different time steps as outputs of different neurons in a deep network. Nevertheless, it has been difficult to train standard RNNs because the gradients either grow or decrease at each time step, so over many time steps they typically explode or vanish (Bengio et al., 1994; Pascanu et al., 2012). While exploding gradient problem can be easily solved by clipping the gradients at some maximum value, the vanishing gradient is effectively addressed by LSTMs due to their design that mitigates this issue (Karpathy et al., 2015).

2.2 Natural Language Generation

Advances in hardware and the increasing quantity of available data have led to significant progress in DL techniques, which have drastically transformed the NLP field in the last decade. As a consequence, NLG systems can be classified as traditional or neural, depending on whether they use deep learning to generate language or not.

2.2.1 Traditional NLG

First works on NLG, which go back to the beginning of the 70s, were based on handcrafted rules. Winograd (1972) describes a computer system that answers questions on a particular domain using simple patterned responses such as “ok.” or “I understand.”, and

slightly more complex responses filling the blank in some predefined sentences. Templates are efficient and allow a control of the the quality of the output by avoiding the generation of ungrammatical structures.

Another approach is the one followed by the automatic novel writer presented in Klein (1973), which uses an English grammar to create murder stories where the motivation, the killer and the victim can vary. The stories are composed of very simple short sentences and they lack fluency, as shown in the example in Table 1, an extract of a story automatically written by this system.

FLORENCE TALKED WITH THE COOK ABOUT THE MURDER.
THE COOK WAS UPSET ABOUT THE MURDER.
JAMES SAID THAT RONALD KILLED DR. HUME.
RONALD DENIED THE ACCUSATION.
RONALD SAID THAT JAMES WAS STUPID.

Table 1: Extract of the output of Klein (1973)

The generation models became more sophisticated in the following years, but continued being fundamentally based on hand-crafted grammars or predefined templates designed by humans (McRoy et al., 2000; Piwek, 2003). These methods are effective when application domains are small and little variation is needed, but they are hard to generalise to other task or domains.

During decades, NLG systems have followed a pipeline architecture. This traditional architecture is divided into different successive tasks, and even though not all systems conform completely to the same stages, there is a consensus regarding the three principal modules described in Figure 5 (Reiter and Dale, 2000).

One of the first and most famous systems that follows the traditional modular architecture is FOG (Goldberg et al., 1994), a bilingual system that generates weather forecast reports in French and English. It is an example of a data-to-text system that takes as input a numerical value of how various meteorological phenomena may vary in a region in a specified time interval, and produces a textual summary of this information after deciding how detailed the information it provides should be. The system was put into commercial use in 1993. Since then, more commercial NLG systems have been developed, such as those offered by the Arria company[2].

Surface realisation, the last standard module that converts text specifications into a correct natural text, addresses issues such as word order determination, inflection or paraphrasing. This stage has triggered specific interest, with recent workshops and shared tasks dedicated to this aspect of the NLG process (*Surface Realization Shared Task 2018* and *Workshop on Multilingual Surface Realization*[3] held at ACL2018). Several attempts have been made to create flexible and domain-independent templates and grammars to perform surface realisation (Mann, 1983; Elhadad and Robin, 1996; Mcroy et al., 2003),

[2]https://www.arria.com/
[3]http://taln.upf.edu/pages/msr2018-ws/

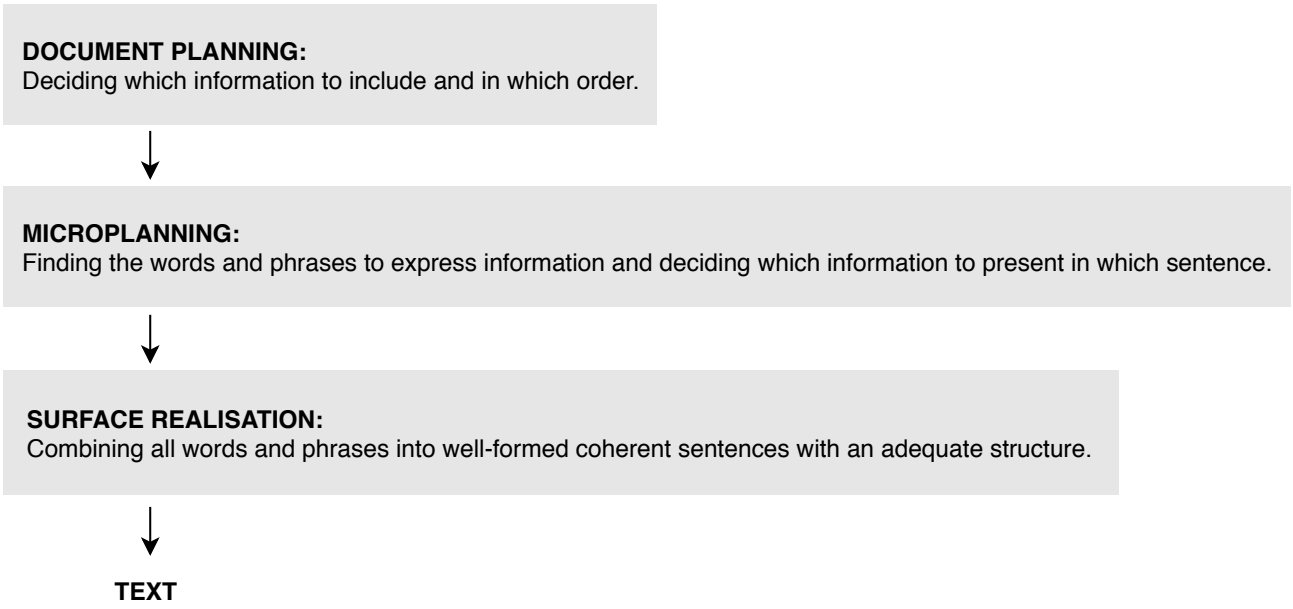

Figure 5: Architecture of traditional NLG systems

but hand-crafting this type of system is often time-consuming, requires expert knowledge of linguistic representation, and the generated outputs still lack the large variations of human-written texts.

As an alternative, statistical systems that do not need such complex rules have been designed. One of the first systems of this type is NITROGEN (Knight and Hatzivassiloglou, 1995), a sentence generator that uses very simple and generic grammars to generate multiple sentences and then selects the best one using a corpus and a statistical model based on bigrams. One of the shortcomings of this system, the lack of syntactic information in the generation process, is addressed in HALOGEN (Langkilde, 2000), a system with a similar approach but which represents the generated sentences as sets of syntactic trees. The tree sets are then classified using a statistical n-grams model.

These traditional approaches are now being replaced with neural NLG and end-to-end data-driven methods, which jointly learn all the stages of NLG directly from data.

2.2.2 Neural NLG

Along with the success of neural approaches in many areas like machine translation or automatic speech recognition, the number of studies that explore different neural architectures for NLG has increased in recent years. DL approaches rely on statistical learning of direct correspondences between inputs and outputs, putting into question the traditional division into modules of NLG systems, and, currently, non-modular end-to-end architectures represent the dominant trend (Gatt and Krahmer, 2018).

Systems that take advantage of DL techniques have demonstrated significant improvements over traditional ones, in particular in terms of generalization to new domains and

scalability, and consequently, NLG has regained weight in the community. Examples the of success of this new paradigm in NLG include its application in dialogue systems (Tran et al., 2017), text summarisation (Chopra et al., 2016), image captioning (Xu et al., 2015) and video description (Yu et al., 2016).

Architectures

Within the advances in DL, the development of RNNs and their variants has been of particular importance for NLG, as they implicitly encode temporal information of sequences of variable size. They have achieved new state-of-the-art results in handling natural texts in areas such as language modelling and automatic speech recognition (Mikolov et al., 2010). Additionally, RNNs are able to model the generation process of text sequences by generating one word at each time step, therefore becoming the dominant paradigm in the field of NLG over the last few years. Examples include Sutskever et al. (2011) and Karpathy and Fei-Fei (2015), which generate texts using two variants of RNNs, i.e., multiplicative and bidirectional RNNs.

Since Graves (2013) proposed to generate sequences using LSTMs, this particular type of RNN has been the most successful and has achieved promising results generating texts in very different domains such as machine translation (Sutskever et al., 2014), video captioning (Venugopalan et al., 2015b) or dialogue systems (Wen et al., 2015).

Most recent NLG models follow the **encoder-decoder** architecture introduced by Sutskever et al. (2014), which has been particularly successful in machine translation and forms the basis of most neural machine translation variants (Bahdanau et al. (2015); Wu et al. (2016)).

Encoder-decoder models are composed of two RNNs. The encoder RNN reads the input data and transforms it into a fixed-length internal representation, from which the decoder RNN generates the desired output. The decoder predicts the next word based not exclusively on the encoded contextual information given by the encoder, but also taking into account all the previously generated words (Figure 6).

This architecture is explored in many NLG studies such as Dušek and Jurcicek (2016) or Nallapati et al. (2016). and is also referred to as sequence-to-sequence architecture.

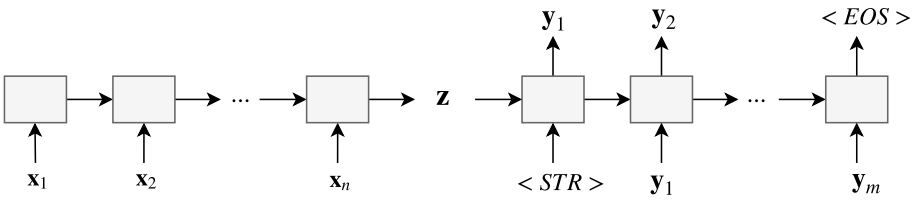

Figure 6: Architecture of a sequence-to-sequence model. $\mathbf{z}$ is the internal representation of the input.

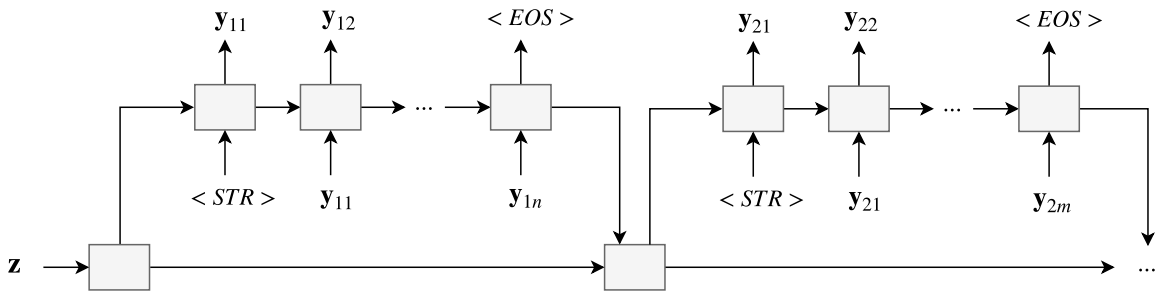

Figure 7: Architecture of a hierarchical decoder. **z** is the internal representation of the contextual information.

However, variations in the architecture of the encoder make these models capable of dealing with non-sequential input, and thus they are appropriate also for data-to-text generation.

Contextual information can be encoded by means of a MLP if it is discrete (Dong et al., 2017), whereas images are typically encoded using a variety of FNN called convolutional neural network (CNN) (LeCun et al., 1999), which has shown impressive results in pattern recognition, by means of a set of filters using convolutions.

A combination of CNN+RNN has been adopted by multiple image and video captioning studies such as Venugopalan et al. (2015a) or Karpathy and Fei-Fei (2015).

Although they can produce coherent sentences, the difficulty of propagating the contextual information along the decoder makes the quality of encoder-decoder models decay as the target sequence length increases (Bengio et al., 1993).

An approach to address the long-term dependency issue is the generative concatenative network introduced by Lipton et al. (2015), which simply concatenates the encoded contextual information with the decoder's input at each step of the generation process. Despite its simplicity, this approach has demonstrated robust results in Costa et al. (2018) and Ni et al. (2017).

A more extended solution is to use **attention mechanisms**. Initially introduced by Bahdanau et al. (2015) to improve automatic translation systems, this mechanism has been applied also in NLG (Mei et al., 2016; Dong et al., 2017) and in addition to dealing with the difficulties of learning long-term dependencies, it learns to focus on specific parts of the input at each step of the decoding. The most relevant parts of the input are thus selected at each time step and then used to feed the decoder.

Challenges for sequential generation models to handle long texts can be alleviated also with **hierarchical decoders** (Sordoni et al., 2015), which generate sentences one by one, instead of generating the whole text at once. As depicted in Figure 7, the decoder is divided into two parts, a sentence level RNN and a more general one, sometimes called paragraph generator. The sentence level RNN works just like a ordinary decoder but each generated sentence is used to update the general RNN, and its output, in turn, is used to feed the sentence level RNN at the next step. In this manner, the system takes into account the

already generated sentences to generate the next one and thus makes it possible to maintain coherence for the whole text without the need to handle one extremely long sequence. The hierarchical structure can be extended also to the decoder and combined with attention as in Li et al. (2015).

Aside from encoder-decoder models, unsupervised learning methods have also been tested recently in the NLG field. Rajeswar et al. (2017) and Yu et al. (2017) for example, make use of **generative adversarial networks** (GAN) to generate texts. These systems consist of two neural networks competing with each other. While a network tries to generate realistic outputs, the other one tries to distinguish the generated outputs from real examples. By training both networks jointly, the generator learns to produce realistic outputs.

Semeniuta et al. (2017) on the other hand, employs a **variational autoencoder** (VAE) to generate tweets. VAEs are recently introduced generative models that in addition to encoding real data and reconstructing it via a neural decoder, are able to learn a probability distribution of encoded real data. New data is then generated by taking samples from this probability distribution and passing it through the decoder. These unsupervised methods must be studied further yet, specially for the generation of longer sequences, since Semeniuta et al. (2017) reported issues to generate texts of more than 50 characters, and Rajeswar et al. (2017) trains its model with sentences of up to 11 words only.

Limitations

Although DL has been a breakthrough in NLG, it still has some strong limitations, such as the issue of out-of-vocabulary (OOV) words.

In domains with very heterogeneous texts, the vocabulary can have more tokens than is feasible to model. One simple approach to deal with this issue is to establish a vocabulary with the most frequent terms and assign all remaining tokens to the special token <UNK>. Although this technique has been often employed, it can lead to a poor performance of the system if the vocabulary is very varied. Different solutions have been proposed to address this problem in other fields such as machine translation, but not all of them are applicable to NLG. For example, substituting each OOV word in the target sentence with its corresponding word in the source sentence (Gülçehre et al., 2016; Luong et al., 2015) is not possible in NLG because there is not a direct correspondence between input and output data.

Another alternative is to generate texts at character level instead of token level, as in Goyal et al. (2016). This approach is computationally efficient because the vocabulary is much smaller and OOVs are totally avoided. It allows the generation of new strings which can be interesting, but can also produce nonsense words and, as a consequence, usually provides worse performance. In addition, the number of steps that are necessary to create a text is much greater at the character level. An intermediate approach between character and token-level generation is the use of word segments of variable size. This can be accomplished via the byte pair encoding (BPE) compression algorithm (Gage, 1994), which splits words of an open vocabulary into learned character sequences within a fixed-

size vocabulary. This method has been used effectively in neural machine translation, for instance Sennrich et al. (2016).

Handling information from the input during the generation process can also be problematic for some neural NLG systems. Generated texts can thus miss important information or introduce again content that has already been mentioned. Specially when constructing long texts or when there are multiple input sources, maintaining coherence and avoiding duplication is challenging for RNNs (Kiddon et al., 2016; Tu et al., 2017). Despite the contribution of attention and gating mechanisms to dynamically control the contribution of contextual data to the generation process, this problem is still far from being solved.

2.2.3 Evaluation

Regarding evaluation, there is no gold-standard metric to measure the naturalness and the quality of the generated texts. In the absence of adequate automatic metrics to automatically evaluate NLG (Reiter and Belz, 2009; Novikova et al., 2017), most NLG systems are evaluated using perplexity (Lipton et al., 2015; Tang et al., 2016) or machine translation metrics like BLEU or METEOR (Kiddon et al., 2016; Dong et al., 2017; Zang and Wan, 2017), in addition to human evaluation.

Some recent studies have suggested the use of referenceless metrics to evaluate NLG in the context of dialogue systems (Dusek et al., 2017; Tao et al., 2018). The main advantage of these metrics based on neural networks is that they learn to measure the quality of the output texts without requiring human ground-truth references, but their use is not quite extended yet.

Since evaluation has become an important topic in NLG, Section 3 is devoted entirely to this subject, with an explanation of the most extended metrics (both human and automatic metrics) with their pros and cons and the influence that the data selected to evaluate the models have on the results.

2.2.4 Automatic Review Generation

The task of generating product reviews has gained attention as an application of NLG, particularly in relation with neural approaches, due to the large quantity of available reviews and their continuous generation.

In addition to data availability, there are other properties that make automatic review generation interesting within NLG. Language changes depending on the particular context in which it is generated (e.g. the language we use changes depending on the sentiments we have in a specific moment). This aspect is an important factor in the generation of natural language and should not be ignored in NLG. When writing a product review, our goal is not only to generate semantically and syntactically coherent texts, but also to express a sentiment with respect to the product or some of its aspects. Since this sentiment is usually related to auxiliary information such as a product rating, this data can be very useful to study the generation of texts based on a predefined values representing a sentiment.

In this scenario, multiple studies centre on generating reviews conditioned on product ratings (Tang et al., 2016) or sentiments (Radford et al., 2017). In addition to affective NLG, stylistic variations with the use of additional attributes such as user identification or product characteristics have also been shown to improve the quality of the generated reviews (Lipton et al., 2015; Dong et al., 2017).

Attribute-conditioned review generation can be integrated into different NLG architectures. While Hu et al. (2017) generate movie reviews via VAEs and control the sentiment and the verbal tense of the generated reviews, Ficler and Goldberg (2017) generate them by means of an LSTM and condition them on six parameters simultaneously. The LSTM is thus fed with a content-related attribute, a sentiment (positive neutral or negative) and four stylistic parameters including the length of the review and whether the writing style is professional or not.

Alternatively, Dong et al. (2017) propose an encoder-decoder model for book review generation. In this case, stylistic conditioning includes the reviewer identifier, while the rating modulates the sentiment of the output. In addition, an identifier of the criticized product helps to determine the content of the review. GANs can be also used to create positive and negative reviews, as in Rajeswar et al. (2017).

One of the biggest challenges in review generation is handling the enormous variety of candidate reviews that satisfy the input attributes. Apart from the explicitly given attributes, there are other factors that influence the generated reviews, making the generation process non-deterministic and the possible results extremely diverse. For example, different users tend to describe different aspects of a product and use different sentiment words to express the same rating score. An adequate evaluation method is crucial to handle this problem and decide which output is correct and which one is not.

Long review generation is also a challenging task due to the aforementioned issues that make it difficult to maintain coherence, though there is little work on this specific issue at present. Lipton et al. (2015) generate short texts for various aspects of the product and join them as a single long review, whereas Zang and Wan (2017) generates long reviews from aspect-sentiment scores, using a hierarchical structure within the encoder-decoder framework.

Finally, another additional issue is that review data feature a significant quantity of proper nouns and slang, which contributes to an enormous vocabulary difficult to deal with.

Recent work on review generation has centred on text generation from fixed attribute vectors, as exemplified by the use of ratings and product identifiers. These contributions have shown that end-to-end DL architectures are suitable for the generation of natural language descriptions from this type of input. However, under these experimental settings, the generation process crucially depends on previously seen items and models are taught to generate text from data similar to that on the training set. As a result, these approaches cannot be evaluated on related but unseen input, such as a new products that may be similar to other previously reviewed items. Such a conditioning makes it difficult to evaluate the generalisation potential of different NLG architectures. The advantages and disadvantages of different types of input data are further discussed in section 4.

Generalisation beyond training data is a challenge for DL methods, which extends to other domains outside of NLG. In image captioning, a method called zero-shot learning learns to recognise new concepts that have not been seen during training, by using a description of them (Romera-Paredes and Torr, 2015). Inspired by this approach, this work tries to exploit unstructured information as product descriptions in order to test the capability of NLG architectures to generalise to unseen products.

3 Evaluation of NLG systems

The evaluation of NLG systems is a central matter in the field, with several studies focused exclusively on this topic (Reiter and Belz, 2009; Novikova et al., 2017). In this section, we describe the core aspects involved for a proper evaluation of NLG systems.

In addition to describing and discussing the adequacy of the evaluation methodology, including the employed techniques and metrics, it is necessary to emphasize the influence of the data we use to evaluate the systems. The results may vary considerably depending on the test data, so it is crucial to select them properly, taking into account the peculiarities of the task and the domain on which the system works. Finally, once the scope and validity of the different techniques and metrics are established, we discuss the use of common evaluation criteria, in order to facilitate comparisons between systems.

3.1 Methods

Evaluation methods are classified into intrinsic and extrinsic methods. While intrinsic methods explicitly evaluate the quality of the generated text by means of different metrics, extrinsic evaluation is dependent on the application domain of the system and measures the output's effectiveness in achieving its specific goal (Gatt and Krahmer, 2018).

An example of extrinsic evaluation for NLG appears in Bartoli et al. (2016), where restaurant reviews are generated, and the system is evaluated by measuring its ability to influence the decision of a user about going or not to a restaurant. Extrinsic or task-based evaluation needs to control all intervening variables in order to carry out the study in a realistic context. For example, Portet et al. (2009) generated summaries from patient data and was evaluated with medical staff making decisions based on the generated summaries. However, this scenario was not completely realistic, since decisions were taken without direct access to the patient, which is a very important source of information in medical decision making.

In addition to an exhaustive control of the variables, extrinsic evaluation is usually highly time consuming and requires a large number of adequate users in order to achieve sufficient statistical power, so it is often considered the most expensive and difficult evaluation method (Carenini and Moore, 2006). Therefore, there is little discussion in the literature about extrinsic evaluation methods for NLG and how they should be carried out.

Intrinsic methods, on the other hand, have been discussed by quite a few studies in the NLG community. These methods measure the performance of a system analysing factors related to text quality, such as correctness of the output or its readability.

This is the approach adopted to evaluate most NLG systems, and almost all review generation systems rely on it. Intrinsic methods are divided on two principal subsets depending on the employed methodology, which can rely on human ratings or on automated metrics based on corpora. In recent years, there has been an increasing interest in corpus-based automatic metrics due to their cheapness, but there is not a consensus regarding their validity and their correlation with human evaluation, and there has thus been a

considerable amount of discussion in the NLG community about their appropriate use.

3.1.1 Human evaluation

In evaluation based on human judgements, different people rate systems' outputs quantitatively on some predefined criteria or compare them with other outputs. Common evaluation criteria are divided in two sets:

- Criteria measuring the linguistic quality of the output (e.g. fluency, readability).

  Examples of studies using these criteria to evaluate the generated texts include Kiddon et al. (2016) and Costa et al. (2018), who measure their outputs by means of grammaticality and readability, respectively.

- Evaluation of the relevance or correctness of the output's content (e.g. accuracy, adequacy).

  In addition to readability, Zang and Wan (2017) evaluate the accuracy of the reviews generated by their system in order to measure how well the review text matches the given input values.

Quantitative evaluation methods help to measure whether an NLG system is working correctly or not, but when the system fails, they do not help to understand why. An advantage of human assessment is that, besides the quantitative evaluation given by ratings, evaluators can be asked to perform a qualitative evaluation and explain why they prefer some outputs over the rest, which provides information about possible improvements for the system.

On the other hand, according to Belz and Reiter (2006) and Reiter and Belz (2009), there are large differences in how different human subjects rate texts, so if the evaluation is conducted with a small number of subjects, the expressed preferences may not be representative of users in general. These differences can be controlled using many subjects, but this can result in an expensive evaluation of the cases when the evaluators must be experts in a certain domain (e.g. medical staff), and it is therefore difficult to execute this type of evaluation.

3.1.2 Automatic evaluation

Automatic evaluation metrics are widespread in many areas of NLP because they are cheaper and faster to run than human evaluation, and thus allow researchers to easily estimate the impact of new algorithms and datasets and to identify weaknesses in a system during the development process.

As described in Section 2, there is currently a lack of consensus regarding NLG-specific evaluation metrics, and automatic metrics drawn from machine translation are frequently employed in the field. Most of these methods measure n-gram overlap between the generated text and human-written references on a corpus. We follow this approach and employ

some of the most popular metrics in machine translation which are BLEU, NIST and METEOR.

- BLEU (Papineni et al., 2002), the standard automatic metric in machine translation, is also the most used automatic metric in NLG. It measures the similarity between the output of the system and one or more ground-truth references in terms of length, word choice and word order. BLEU counts n-gram overlaps, (position independent) between a candidate sentence and the references using a modified version of precision and gives a score between 0 and 100, with reference sentences scoring 100. It is standardly used with n-grams up to order 4 (BLEU-4), which obtains best correlation with human judgements evaluating machine translation systems.

  BLEU is usually computed at corpus-level, because the metric is based on the geometric mean of different n-gram order matches, making it less reliable at the sentence level where missing higher order matches result in a score of 0.

- The NIST score (Doddington, 2002) is based on BLEU but differs from it in its use of the arithmetic mean and its emphasis on lower frequency n-grams, yielding a metric where more informative words have a higher impact on the final score. While BLEU uses the geometric mean of the n-gram precisions to increase the influence of the larger n-grams, NIST uses the arithmetic mean and its brevity penalty is less sensitive to the impact of small length variations.

- An important development of automatic metrics is the use of semantic resources beyond literal n-gram overlap, as in the case of METEOR (Lavie and Agarwal, 2007), a metric which notably includes stem, synonym and paraphrase matching to account for surface variations not accounted for by strictly word-based metrics. METEOR gives a sentence-level similarity score based on unigram recall and precision, where matches in content words have more influence.

All the automatic metrics, however, must be validated analysing how well they correlate with human preferences. Therefore, multiple studies have examined correlations between human judgements and automatic evaluation metrics in NLG, with varying results:

Belz and Gatt (2008) found that there was no significant correlation between the results of an extrinsic evaluation and metrics like BLEU and NIST in a referring-expression generation task. Reiter and Belz (2009) on the contrary, conclude that these two metrics, and specially NIST, are able to measure the linguistic quality of generated texts in the domain of weather forecasts. However, this study suggests that the analysed metrics do not provide a useful measure of content quality, because none of them exhibits a significant correlation with human accuracy judgements.

In the machine translation field, NIST for unigrams correlates better than BLEU with human judgements of adequacy, and METEOR consistently demonstrates a high correlation with various human judgements in independent evaluations such as thos performed in EMNLP WMT 2011 (Denkowski and Lavie, 2011).

"I loved the idea of a person who tries to murder but is continuously unsuccessful. I also like the twists and turns and the surprise at the end. But I did think of a better ending..."
"I truly enjoy most of Mr. Wilde's writing. I enjoyed reading this. I have read most of Oscar's stories and find most of them quite a good read."

Table 2: Examples of positive human reviews of the same book in the dataset

Novikova et al. (2017) suggest that automatic evaluation metrics can help detect cases where the system performs poorly and Reiter and Belz (2009) confirm that they can be reliable at system-level. According to the latter, although results are data- and system-specific, automatic metrics facilitate the development process by offering a way to easily evaluate the impact of small changes in systems and algorithms.

3.2 Impact of Test Data

One substantial difference between the machine translation and NLG tasks is worth noting, as it impacts the use of the aforementioned metrics. In machine translation, the references used as ground-truth and the generated translation should contain the same information, even though they can use different words to express it, and this information has been given as input in the source language. In many NLG tasks, however, and particularly in review generation, the information that can appear in the output texts is not restricted in that manner.

Thus, one of the major problems when automatically evaluating a review generation system is that two reviews expressing an opinion about the same product may have little to nothing in common even when they share the same rating, as shown in the examples of Table 2. Therefore, the range of possible outputs is wide, and when evaluating results on review generation, the ground-truth references used to evaluate the output acquire a significant importance.

Although vocabulary variability is preferred by many human evaluators, automatic evaluation metrics assign high ratings to systems that maximise the likelihood of the corpus generating outputs with the highest frequency and penalises those that tend to use different words and phrases. This problem is addressed in Novikova et al. (2017), where a grammatically correct output that expresses the intended meaning well, and which indeed is very highly rated by human evaluators, obtains a very low score from the automatic metrics described above, because the generated text has low overlap with the reference to which it is compared (Table 3).

Using a high-quality test set with multiple references can provide reasonable coverage of possible variations and mitigate this issue. in this work, we follow this approach by providing a minimum number of four references for the test set when we employ the automatic word-based metrics to identify relative weaknesses and key strengths of different approaches.

INPUT: inform(name = the donatello, hasinternet = yes) OUTPUT: “well there is a hotel with internet access called the donatello” REFERENCE: “the donatello has internet”
AUTOMATIC EVAL.: **1.4**/6 HUMAN EVAL.: **6**/6

Table 3: Example of disagreement between automatic and human evaluation, from Novikova et al. (2017)

Obtaining good reference texts can be expensive, specially in domains where texts must be written by experts, and for this reason, referenceless metrics are gaining interest in NLG (Novikova et al., 2017; Dusek et al., 2017). In addition to n-gram-based metrics, we will therefore also include results in terms of perplexity of the generated sentences to measure output fluency without requiring references. Perplexity measures how well a language model trained over the training set predicts the generated texts, where low values indicate that the probability distribution of the language model is good at predicting the sample.

The similarity between the data used to train and to test the system also influences the results of the NLG systems. Elliott and de Vries (2015) create image descriptions, and their results notably change depending on the data they use for evaluation, with better performance on images depicting people performing actions than in a more varied dataset, where the training data result less useful. To take this aspect into account in our evaluations, we will also include a perplexity-based split of the test set to evaluate the results on reviews of varying complexity.

4 Structured and Unstructured Input

By definition, the output of NLG is always a text, but the input can vary considerably from one system to another. This variability, which does not exist in other fields of NLP where the source of information is always the same (e.g., textual input for natural language understanding or audio files for automatic speech recognition), becomes visible in the classification of NLG systems, which are divided, depending on the type of input, in data-to-text and text-to-text systems. This is mainly due to the wide range of tasks for which NLG is used, since it is evident that the same type of input cannot be used for image captioning and for automatic text summarisation, but even within the same task, different approaches tend to represent the same source data in different ways.

Although this topic has generated interest for a long time, and there are related studies since the beginning of the 90's (McDonald, 1993), the ideal input format for NLG is still undetermined. Each system tries to express the input in a format that is practical to its specific application and it is hard to define a general-purpose representation of data.

The importance of this issue is that the nature of the data directly influences the type of architectures to be used, and therefore, produces an impact on the ultimate results achieved by the system. As a consequence, a good selection of the input type can lead to improvements on a given task. Additionally, being able to deal with different representations of the data can make it easier to reuse different DL methods across systems.

The most common input data consist of numerical data or categorical features that can be easily expressed by means of a fixed-size vector (e.g., a one-hot vector expressing a class). In NLG for dialogue systems, the input is usually a meaning or semantic representation of the dialogue act (Wen et al., 2015) and recently, the growing interest in image and video captioning has made the task of generating text from visual input an important task within NLG. All these types of inputs can be divided into structured and unstructured data, each with its advantages and disadvantages, where the type of task is decisive.

4.1 Structured Input

With structured data, each of the features has a well-defined meaning and structure. The price of a product or the age of a person, for example, can be expressed with simple structured input features, via a number that takes real and discrete values, respectively. The data structure used for the task can be a discrete variable (e.g., movie rating) or a continuous variable (e.g., temperature). Class membership can also be represented using a fixed-size vector and one-hot encoding. For instance, a vector of size 2 can express the gender of a person or indicate if a sentiment is positive or negative.

In automatic review generation, it is very common to use one-hot encoding to feed the system with a discrete variable like the rating of the reviewed product or the user or product identifier (Tang et al., 2016). In fact, besides review generation, a large amount of affective NLG work relies on human ratings. This type of data is easy to obtain and cheap, does not require hard preprocessing and is also easy to understand.

On the other hand, this type of input makes it difficult to generalise to new instances.

For example, in order to train a review generator with reviews of 100 different products, we can use a vector of size 100 and one-hot representation to feed product information to the system. But, what if after training the model, we want to generate a review about a new product? We cannot express this product by means of the input vector, because it would then be a vector composed entirely of zeros and therefore, with no associated information. A possible solution is to feed the model with different characteristics of the products. In this manner, the model would be able to learn the relation between some specific characteristics of the products and the output text, and we could therefore create a review for a new product, unseen during training, provided we have the appropriate information about it. These characteristics, however, would need to be defined before training.

Another example of structured data is semantic or meaning representation. Table 4 shows an example of this type of input in the restaurant review domain. As we can see, it is also very simple to understand, but it requires more preprocessing than raw numerical data.

MR:	name[The Wrestlers] priceRange[cheap] customerRating[low]
NL:	The Wrestlers offers competitive prices, but isn't rated highly by customers.

Table 4: Example of meaning representation (MR) and its corresponding natural language output (NL), from Novikova et al. (2016)

Abstract Meaning Representation (AMR) is a semantic representation language introduced by Banarescu et al. (2013), widely used as input for NLG (e.g. Pourdamghani et al. (2016); Flanigan et al. (2016); May and Priyadarshi (2017)). Using rooted, labelled, directed, acyclic graphs, AMR aims to abstract away from syntax and represent sentences according to their meaning. The graph notation, which is very useful for computer processing, is adapted to the PENMAN notation[4] for human reading and writing.

This type of input, as well as logical formulae that can also be used as input in NLG (El Ghali, 2002), expresses the intended meaning of the desired text. This means that sentences with similar meaning have the same semantic representation, even if they are worded in a different way. Intuitively, it seems very logical to use this type of data as input for NLG, but it is less interesting if one objective of the system is learning to decide which part of the data is important and which one is not before writing the text.

[4]https://www.isi.edu/natural-language/penman/penman.html

When generating reports of financial data for example, the system must select which part of a large quantity of data can be useful to the user, and use only this information to generate the output. This step corresponds to the first stage of the traditional NLG architecture, but if the production of text starts from a meaning representation, this part of document planning is already done, and the system only covers the micro-planning and final surface realisation tasks (McDonald, 1993). Thus, this approach, which has proved very useful for dialogue systems, is less helpful if the objective is to use NLG to create summaries of data that are difficult to understand by people.

4.2 Unstructured Input

Unstructured data refers to different types of data like raw audio, images, or text of variable size with an open vocabulary. In the NLG domain, the latter two are the most common examples of this type of input and possible features are the words in a piece of text or the pixel values in an image (Venugopalan et al., 2015a; Chopra et al., 2016; Karpathy and Fei-Fei, 2015).

While humans are really good at interpreting unstructured data (we easily understand audio files as well as images and texts), computers are more suited to dealing with structured input. Despite the incredible success of DL exploiting structured data with, for example, its ability to make accurate predictions from giant databases that are almost impossible to understand by people, successes on unstructured data seem to be more impressive. Perhaps because of the natural ability of humans to understand this type of data, people are often surprised that a neural network can recognize an image or understand a text. These advances can give the impression that computers really can "understand" things as humans do, and are assigned human capabilities when in reality their achievement is the discovery of patterns in data via mathematical calculations.

The principal reason behind the necessity of exploring NLG architectures that are able to exploit unstructured input is that such data is necessary for text-to-text NLG tasks such as automatic summarisation. Additionally, unstructured input can be useful also in data-to-text generation, providing the possibility, for example, of generalising to new inputs. Going back to the example of generating reviews of previously unseen products, the model can be trained with textual descriptions of the products instead of using an identifier or some characteristics defined a priori. Using this approach, the model should be able to create a review about a new product by taking advantage of its description, even if the product has not been seen during training.

Moreover, the use of unstructured data like texts makes it possible to evaluate some techniques that have been proved useful in other domains. This is the case of attention mechanisms, which are applied in systems with sequential input such as texts, and are an essential component of current machine translation systems.

4.3 An Unstructured Input Task for NLG

As previously mentioned, most recent work in review generation is based on fixed attribute vectors. Consequently, the task of exploiting unstructured data to test the generalisation capability of NLG systems to similar but different items has not yet been addressed. One of the principal objectives of this work is to fill this gap, by providing the means to evaluate the ability of various approaches to exploit latent information in unstructured data for the generation of reviews of unseen products.

To realise this objective, the proposed task provides as input both product descriptions and user ratings, excluding other possible inputs such as product identifiers and review titles, as conditioning on the former would prevent generalising to unseen products and using the latter would not provide input-level information independently of the content of the reviews.

Note that the use of unstructured contextual information could be extended to other aspects of interest, such as information on other viewed products, for instance. We focus on a variant of the task based on product descriptions as they provide the largest amount of textual semantic information in the datasets and thus allow for a first evaluation on the impact of unstructured contextual information for review generation. We expect the architectures to be able to extract valuable information from the product description and to generate reviews that cover the topics derived from it, expressing at the same time a certain sentiment indicated by the input rating.

A dataset has been prepared with the aim of testing different DL architectures in this task. The structure of the corpus has been explicitly designed for this task and its partition allows for a robust evaluation of the results.

4.3.1 Dataset preparation

	#reviews	Review length			#products	Description length		
		min	max	average		min	max	average
TRAIN	747513	25	97	47.24	30924	25	150	78.24
DEV	99668	25	95	49.03	8696	25	150	77.08
TEST	111876	25	98	49.18	12804	25	150	77.56
TOTAL	959057	25	98	47.65	52424	25	150	77.85

Table 5: Statistics of the dataset

The dataset used to perform the experiments in this work is a subset of the Amazon Product Review corpus (McAuley et al., 2015). While the original dataset contains millions of reviews of various products, this subset contains only book reviews and is composed of about 960K reports, which refer to more than 52K books. The subset is prepared for our specific task, each review being paired with the corresponding description of the product and a rating that expresses the user's feeling towards the product by means of a score ranging from 1 to 5 (from negative to positive sentiment).

Noisy instances have been removed from the dataset by selecting only those reviews and descriptions where at least 50% of the words were in English. All texts have also been filtered so that they have a delimited length (between 25-100 words for reviews and 25-150 words for descriptions).

The selected data have been split into three different subsets: the TRAIN set, which contains the instances that are used to train the system; the DEV or development set, which is used to monitor the progress of the system while training and to select the optimal model; and the TEST set, to evaluate the final model.

In order to adequate this corpus to the task of measuring the ability of a system to generalise to unseen products, each product description appears only in one of the sets, i.e. there is no product overlap between the partitions. Additionally, all entries in the test set have at least four references for each generated review, in order to perform a strong evaluation that takes variability into account.

Table 5 describes the corpus in terms of number of reviews, description length and review length[5].

Since the data is notably unbalanced[6], and with the goal of performing a consistent evaluation, we attempted to maintain the same distribution of ratings over the three sets of the corpus. However, this proved difficult due to the limited number of negative reviews, as most of them have less than four references and therefore cannot be part of the test set. Despite some differences between the sets, it has nonetheless been possible for all the ratings to be represented in every set (see Table 6). Note that even a small percentage like 0.40% represents hundreds of reviews due to the large size of the corpus.

	R=1	R=2	R=3	R=4	R= 5
TRAIN	3.95%	3.59%	7.42%	18.58%	66.45%
DEV	4.36%	3.53%	6.95%	17.57%	67.59%
TEST	1.74%	0.40%	1.82%	14.54%	81.50%

Table 6: Distribution of ratings over different sets

Finally, the test set was further split into three additional subsets, based on perplexity scores, with the aim of having test sets of varying complexity: TEST-LOW-PPL , TEST-MEDIUM-PPL and TEST-HIGH-PPL. A 5-gram language model was first trained with the KenLM toolkit (Heafield, 2011) over all the reviews in the TRAIN set. Based on this language model, all the reviews in the TEST partition were then sorted according to their perplexity and divided into three equally numbered groups. The reviews were again filtered in order to have at least four references for each output in every group. The main goal of this new partition was to evaluate the results in a more fine-grained manner, as simpler and shorter reviews, with lower perplexity, might be less useful for instance to compare different architectures. Table 7 describes the perplexity-based test sets in terms of number of reviews, number of products and perplexity scores.

[5]The length of the texts indicates the number of words before tokenisation.

[6]More than 80% of the ratings have a score of 4 or higher.

	# reviews	# products	ppl
TEST-LOW-PPL	19951	3814	24.12
TEST-MEDIUM-PPL	17191	3594	57.12
TEST-HIGH-PPL	19191	3772	152.20

Table 7: Test set size of low/medium/high perplexity

The prepared datasets thus support our established goals of providing unstructured input information which can be used for a fine-grained comparison of different NLG architectures.

5 Neural NLG Architectures

This section describes the neural network architectures employed in the experimentation in Section 6. Each one of the following four variants was meant to test different ways of exploiting product descriptions: **LSTM-LSTM**, **LSTM-Att**, **LSTM-Conc** and **LDA-Conc**. An additional architecture, **productID**, that uses the product identifier instead of the description, was defined as baseline throughout the experiments.

The presented DL architectures are all variations of the basic encoder-decoder model, and the decoder is in all the cases formed by two recurrent layers with LSTM units (equations 9-13).

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \tag{9}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \tag{10}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{(t-1)} + \mathbf{b}_o) \tag{11}$$

$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{(t-1)} + \mathbf{i}_t \tanh(\mathbf{W}_{ig}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{(t-1)} + \mathbf{b}_g) \tag{12}$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t) \tag{13}$$

The decoder predicts the following word of the review using the information about the product, the rating, and all the previous words of the review.

As the goal of these experiments is to explore different ways to exploit the non-structured data provided by item descriptions, the manner in which the decoder is fed with the current word and the rating attribute (structured data) remains the same across all architectural variants. At each time step of the decoder, the rating, which is represented by the one-hot vector $\mathbf{r}$, is concatenated to the embedding of the current word and to the product information given by the encoder.

We describe below the three essential components common to all variants.

Softmax

A softmax layer stacked over the decoder outputs the probability of being the next generated item for each word in the vocabulary.

$$Softmax_i((x_1, \ldots, x_n)) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{14}$$

The softmax function is defined by equation 14 and it is very commonly used for multi-class classification tasks. For a k-dimensional vector, it returns k values between 0 and 1 where all outputs sum to 1, so they can be seen as the probabilities of k different categories. To obtain a probability for each word of the vocabulary, the decoder's output is resized to the size of the vocabulary before introducing the values to the softmax layer.

Drop-out

All the architectures have a special layer called drop-out between the two layers of the decoder. Dropout layers (Hinton et al., 2012) randomly annul some of the elements of the input in order to prevent overfitting, an issue typically present in large neural networks with good results obtained on the training data but poor ones on test data. Overfitting is greatly reduced by randomly omitting some of the neurons in the network on each training instance, with large improvements obtained on many DL tasks like speech and object recognition.

Word embeddings

As in most DL architectures that deal with textual data, the tokens in both the descriptions and the reviews are represented by a one-hot vector which is then transformed using an embedding layer. Embedding vectors represent the semantics of a word using dense vectors that take continuous values, instead of using sparse vectors as in one-hot encoding. In addition to taking advantage of the semantics of the words, where words with similar meanings have a similar representation, this representation reduces the size of the vectors, which goes from being equal to the size of the vocabulary (usually thousands of units), to having a pre-determined smaller size. These new vectors, which are used to feed the encoder and decoder, are calculated by a linear layer that is trained along with the rest of the network.[7].

The description of the product is represented by the sequence of embedding vectors $D = (\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_M})$, and $\mathbf{w_i}$ denotes the embedding of w_i, the i^{st} word of the review. <STR> and <EOS> are special tokens that are added to all reviews to indicate the beginning and the end of the review, respectively.

While the base architecture is the same for all variants, differences between the description encoding mechanism and the decoder architectures produce different models. We now turn to describe each architectural variant in detail.

5.1 ProductID

This is the simpler architecture, meant to provide a baseline upon which to measure the usefulness of textual descriptions for review generation, in the scenario of generalising to new products. For this reason, it does not exploit any product-related latent information from the description and instead of using the item description D, each product is represented by an identifier, using a unique one-hot vector $\mathbf{p}_{id}$. This vector is passed through an embedding layer and concatenated to the current word and the rating (Figure 8) at each time step, in order to predict the next word.

[7]The embedding layers used to encode the words and the softmax functions do not appear in the schemas of the architectures since the objective of these figures is to highlight the differences between the different variants and these layers are the same in all cases

Due to the nature of the task, with no overlapping items between the partitions of the corpus, the size of the one-hot vector is equal to the number of products in the training set. Therefore, when introducing a new product, the corresponding $\mathbf{p_{id}}$ vector is composed of only zeros and consequently does not provide any information about the product. Thus this variant will learn the most common expressions according only to the given user rating.

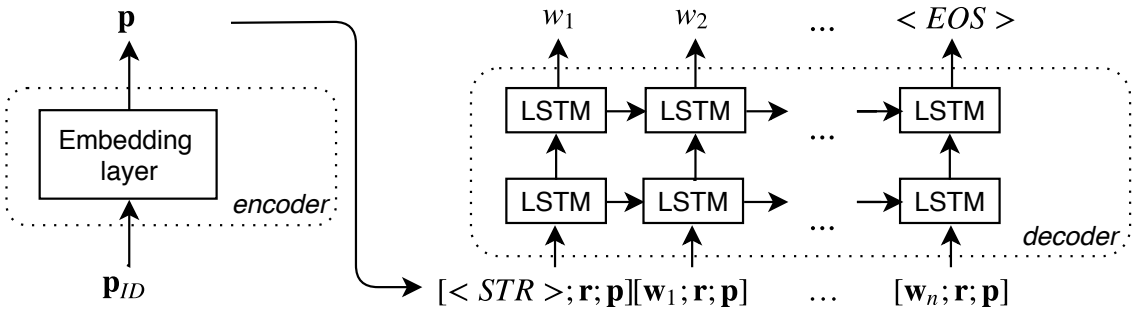

Figure 8: ProductID architecture

The results obtained with this architecture are used as a baseline and compared to the ones obtained using the description of the product, in order to determine whether the latter contributes to generalising to new products.

5.2 LSTM-LSTM

The second approach leverages the description of the product using a standard encoder-decoder model as depicted in Figure 9.

The encoder is a single LSTM layer that transforms the description $D = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_M)$ into a new sequence $h^D = (\mathbf{h}_1^D, \mathbf{h}_2^D, \ldots, \mathbf{h}_M^D)$.

In this architecture, the hidden vector of the decoder is initialised with the last output $\mathbf{h}_M^D$ of the encoder, which encodes the information of the whole description. The decoder thus generates a text according to the input rating, but with its internal parameters conditioned also on the description of the reviewed product.

If the number of units is not equal in both the encoder and the decoder, an additional layer will be needed to resize the vector $\mathbf{h}_M^D$ and make it possible to use it as the initial hidden vector in the decoder. This transformation is performed by a simple linear layer.

Since LSTMs have some limitations regarding the propagation of the input over the decoder, particularly with long sequences, which are numerous in the corpus we use, two additional variants were built and evaluated, as described in the next two sections.

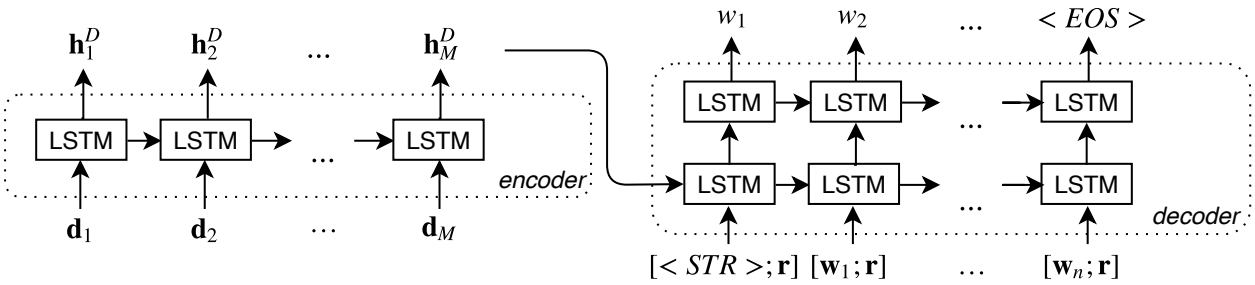

Figure 9: LSTM-LSTM architecture

5.3 LSTM-Att

LSTM-Att uses the attention mechanism introduced by Bahdanau et al. (2015) to address the problems of RNNs to propagate the information over long sequences. This model can attend to different parts of the sequence h^D at each decoding step, inducing a soft alignment model that scores how well the inputs around position j and the output at position i match.

This is carried out by computing, at each decoding step, the context vector $\mathbf{c}$, a weighted sum of the decoder's outputs, via equations 15-18, where $\mathbf{h}_j$ is the j^{st} output of the encoder and $\mathbf{s}_t$ the hidden vector of the decoder at time t.[8]

$$\mathbf{c}_t = \sum_{j=1}^{M} \alpha_{tj} \mathbf{h}_j^D \tag{15}$$

$$\alpha_{tj} = Softmax_j((e_{t1}, \ldots, e_{tM})), \quad \forall j = 1, \ldots, M \tag{16}$$

$$e_{tj} = \mathbf{W}_{oe} \tanh(\mathbf{W}_{se}\mathbf{s}_{t-1} + \mathbf{W}_{he}\mathbf{h}_j^D), \quad \forall j = 1, \ldots, M \tag{17}$$

$$s_0 = \mathbf{W}_{he}\mathbf{h}_M^D \tag{18}$$

Once the context vector is calculated, it is concatenated to the rating and to the previous word forming the vector $[\mathbf{w}_{t-1}; \mathbf{r}; \mathbf{c}_t]$ which is the input of the decoder. Finally, the current hidden vector of the decoder is employed together with the context vector, the previous word and the rating of the review to predict the next word of the review via equation 19, where v_k is the k^{st} word of the vocabulary.

$$p(w_t = v_k | w_1, \ldots, w_{t-1}, D, \mathbf{r}) = Softmax_k\left[\mathbf{W}_{op}(\mathbf{W}_{sp}\mathbf{s}_t + \mathbf{W}_{cp}\mathbf{c}_t + \mathbf{W}_{xp}[\mathbf{w}_{t-1}; \mathbf{r}])\right] \tag{19}$$

[8] As in LSTM-LSTM, the hidden vector of the decoder is initialized using the last output of the encoder

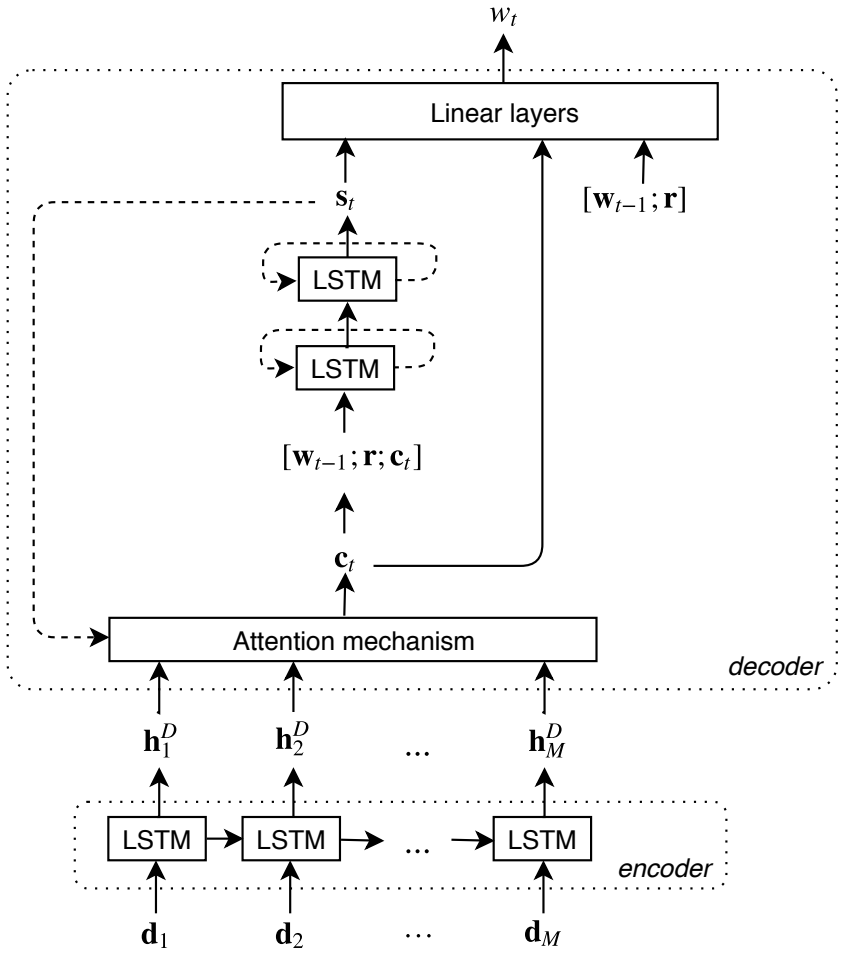

Figure 10: LSTM-Att architecture - Prediction of w_t

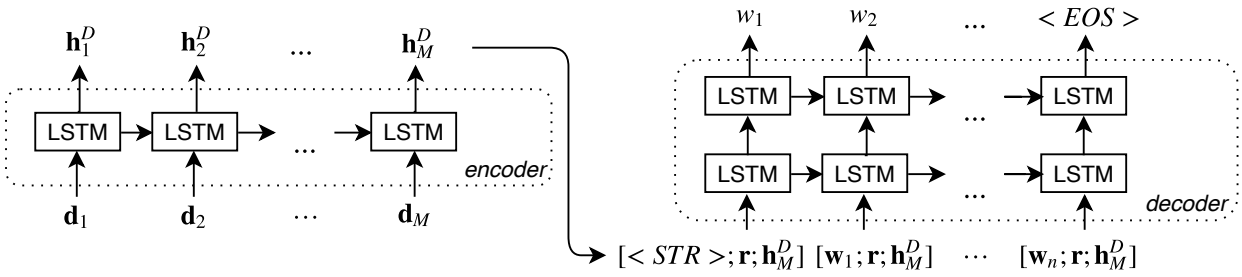

Figure 11: LSTM-Conc architecture

5.4 LSTM-Conc

This architecture, unlike LSTM-Att, uses a Generative Concatenative Network (Lipton et al., 2015) to avoid the long-range dependency problem. In this approach, which is much simpler than the attention mechanisms, the encoded description is concatenated to the input of the decoder at each step of the generation process.

As shown by Figure 11, the last output of the encoder, $\mathbf{h}_M^D$, is transmitted to the decoder together with the one-hot vector of the rating $\mathbf{r}$ and the embedding vector of the previous word, $\mathbf{w}_{t-1}$, to predict the next word w_t.

This variant ensures the propagation of the input along the decoder. However, it does not focus its attention on different parts of the input at each step as attention mechanisms do, but rather uses the entire information of the encoder as it is passed to the decoder at each time step.

5.5 LDA-Conc

Finally, LDA-Conc provides a different way of encoding the description of the product. Instead of using a RNN, this variant makes use of Latent Dirichlet Allocation (Blei et al., 2003) to encode the unstructured information of the description.

Latent Dirichlet Allocation (LDA) is a generative probabilistic model for collections of discrete data such as text corpora. Having a vocabulary and a predefined number of topics K, LDA selects a topic mixture for each document by sampling from a Dirichlet distribution over the K topics. Then, each word of the document is generated by sampling a topic from this mixture and picking a word according to a multinomial distribution conditioned on the selected topic.

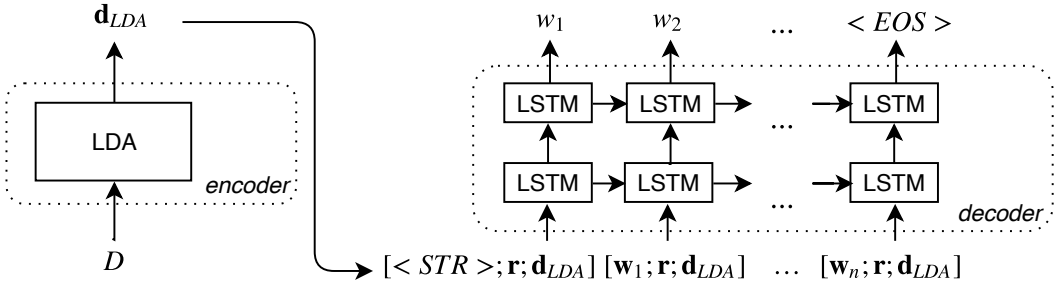

Figure 12: LDA-LSTM architecture

Assuming that a corpus is generated using this process, LDA can infer the topic representation of each document and the words associated to each topic. Computing the desired distribution is intractable, but it can be approximated using simpler distributions via different methods like collapsed Gibbs sampling (Xiao and Stibor, 2010) or the variational Bayes algorithm (Hoffman et al., 2010). Thus, LDA is able to extract the topics of

a text and represent it by means of a vector that indicates in what amount these topics are present in the text, which can be understood as a probability distribution over a fixed set of topics.

In the LDA-Conc architecture, an LDA model has been trained on all the descriptions of the training set. The inferred parameters have been used to encode the input description D into the fixed-sized vector $\mathbf{d}_{LDA}$ that represents its topics. This vector is resized so that its minimum component is always 0 and the maximum 1, concatenated to the rating and added directly as input to the decoder as depicted in Figure 12.

6 Experiments

A set of experiments was prepared to evaluate the different architectures on the task. In this section, we describe the data preprocessing steps, the selection of hyperparameters, and the procedure to train the models.

6.1 Data Preprocessing

Before training the neural networks, a preprocessing of the data is required to reduce input data sparsity and normalise the sentences with respect to casing and tokenisation.

Reviews and descriptions were thus lowercased and tokenised. All numbers were substituted with the special token <num>, while punctuation marks and stopwords were removed from the descriptions. Additionally, <STR> and <EOS> were added to indicate the beginning and end of each text in the corpus.

The large size of the vocabulary means that selecting the most frequent tokens and replacing the others with a unique special token is not optimal. Instead, we opted to address the OOV word problem using the approach presented in Sennrich et al. (2016), which is based on the Byte Pair Encoding compression (BPE) algorithm described in Gage (1994).

BPE is a simple method for data compression, where the most common pair of consecutive bytes of data is replaced with a byte that does not occur within the data. Sennrich et al. (2016) noted that this method could enable neural machine translation systems to operate with an open vocabulary, even though the systems require a fixed input vocabulary. Words of low frequency, such as proper names, can thus be segmented after learning an optimal compression on a training corpus.

BPE can be applied in two ways: learning one encoding for the source language and another for the target language, or learning a single encoding on the union of the two vocabularies. While the usage of two independent encodings creates more compact texts and vocabularies, it may segment the same word differently in the two languages and insert or delete characters when copying names. Learning BPE jointly on the concatenation of the two languages reduces this problem and is therefore effective in tasks where source and target texts share names that can simply be copied into the output text.

This procedure could be used for our NLG task, where descriptions and reviews share numerous proper names such as authors' or characters' names. We therefore followed this approach and calculated BPE jointly on descriptions and reviews. 30K BPE operations were executed, which resulted in $\sim$ 30K instances in the new vocabulary (number of merge operations plus the character vocabulary).

We went further and reduced the vocabulary to those symbols that appear in the training set, since they also cover all the vocabulary in DEV and the number of components is reduced considerably, particularly for reviews (see Table 8).

Table 9 shows some examples of how infrequent terms of the descriptions are split after applying BPE.

	# tokens	# segments (after BPE)
DESCRIPTIONS	83734	27523
REVIEWS	305232	19266

Table 8: Vocabulary of TRAIN set

Original		After BPE
embarrasment	→	embarrass/ment
sacrament	→	sacra/ment
sacraments	→	sacra/ments
republished	→	re/published
instrumentalist	→	instrum/en/ta/list

Table 9: Words split by BPE

6.2 Model Training

The architectures are trained using mini-batches of 50 instances to speed up convergence and the employed error measure is **cross entropy** (equation 20). This measure is widely used as loss function when the output can be understood as a probability distribution, as is the case with neural networks which have softmax activations in the output layer.

$$H(\hat{y}, y) = -\sum_{i} \hat{y}_i log(y_i) \tag{20}$$

To perform gradient descent, the **Adam** method is used (Kingma and Ba, 2015), which computes individual learning rates for different parameters instead of using the same to update all the weights.

6.3 Hyperparameters

The learning rate is initialized at 0.001 and it decays at each epoch using a multiplicative factor of 0.998. All the gradients are clipped at a threshold of 5.

Hyperparameters related to the number of layers and their size are summarised in Table 10.[9] The embedding layer, through which all the tokens pass before going to the LSTMs, is composed of 200 units.

6.3.1 Number of LDA topics

One of the most important parameters in LDA models is the number of topics. Although there are some methods to select the best number of topics (Griffiths and Steyvers, 2004; Arun et al., 2010; Deveaud et al., 2014), selection is almost always performed intuitively, depending on the characteristics of the data.

[9] In the table, "att." refers to the attention mechanism specified by equations 15-18.

	Encoder		Decoder		
	Layers	# units/layer	Layers	# units/layer	Drop-out
ProductID	1 embedding	200	2 LSTM	512	20%
LSTM-LSTM	1 LSTM	256	2 LSTM	512	20%
LSTM-Att	1 LSTM	256	2 LSTM (+att.)	512	20%
LSTM-Conc	1 LSTM	256	2 LSTM	512	20%
LDA-Conc	LDA model	144 topics	2 LSTM	512	20%

Table 10: Hyperparameters of all the architectural variants

The heuristic method used in this work to select the number of topics for the LDA-Conc architecture is based on finding the maximum amount of topics that are linearly independent in the training set.[10] This was done by means of grid search, analysing the topic-word matrices of various LDA models constructed over the product descriptions in TRAIN.

After training the models with different quantity of topics between 100 and 200, the singular value decomposition (SVD) of the topic-word matrices is then used to check whether the extracted topics are linearly independent or not.

The number of non-zero singular values of a matrix is equal to the rank of the matrix, which is also the number of linearly independent rows. Since singular values are positive, the simple verification that the minimum singular value is different than zero establishes that the extracted items are linearly independent.

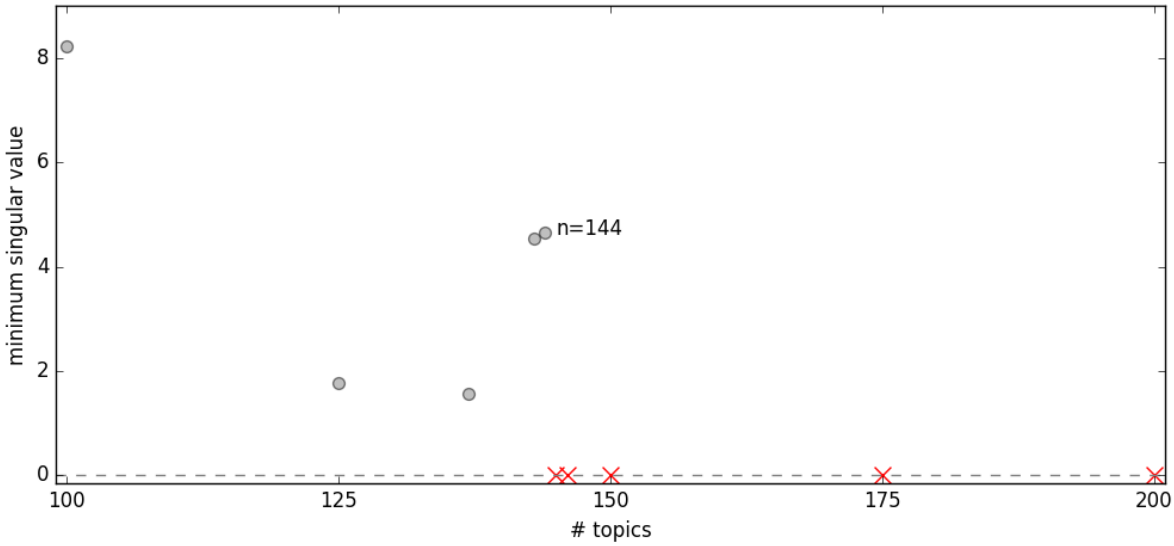

Figure 13: Minimum singular values for different number of topics

The results on our dataset suggested that the maximum number of linearly independent topics was 144 (see Figure 13), and the corresponding model was used to encode the

[10] This idea was suggested to us by Manex Serras (*personnal communication*).

descriptions (note that the results may change depending on the learning algorithm and other hyperparameters selected to train LDA).

Examples of the dominant topics of two training descriptions extracted using 144 topics are shown in Table 11.

Description	Topics
"Panurat Poladitmontri, chef-owner of the Lemon Grass Grill Restaurant in Seattle, is proud of his Thai heritage. Recognized as one of the leading Thai chefs in America, he has authored cookbooks..."	t88, t118
"Ian Cinnamon is a 15-year old phenom who has been programming for over 7 years, and is certified in both Java and C++. He is currently a sophomore at Harvard-Westlake School."	t17, t88
t17: university, health, professor, medical, research, center, medicine, dr, school, nutrition. t88: writing, first, books, author, lives, years, also, two, children, time. t118: food, cook, chef, kitchen, andrew, specializing, low, culinary, cookbooks, dishes.	

Table 11: Dominant topics in each description and the top 10 keywords of the topics

6.3.2 Temperature

The output of the decoder is the probability distribution of the words in the vocabulary, so the most straightforward way to generate reviews is to maximize the likelihood by selecting the most probable word. However, under this approach, the generated texts tend to be very conservative and repetitive. On the other hand, using random sampling to select a word depending on its probability can result in chaotic and incoherent texts.

We addressed this problem via temperature (Hinton et al., 2015), a hyperparameter that is meant to control the diversity of the predictions made by random sampling.

$$p(w_t = v_k|w_1, \ldots, w_{t-1}) = Softmax_k(h_t/\tau) \tag{21}$$

Temperature (τ) divides the non-normalised probability distribution returned by the last layer of the decoder before applying the softmax activation function (equation 21). This division by a value in the range $(0, 1)$ increases the probability of the most likely words, thus making random sampling (RS) more conservative, but still with more variability than maximum likelihood (ML). Within this range, temperatures closer to 0 imply more conservative predictions, while higher temperatures will imply more variability in the generated words. Figure 14 shows how probabilities change using different values for temperature.

After training all the architectural variants, the results obtained on the development set will be used to select the best value for temperature.

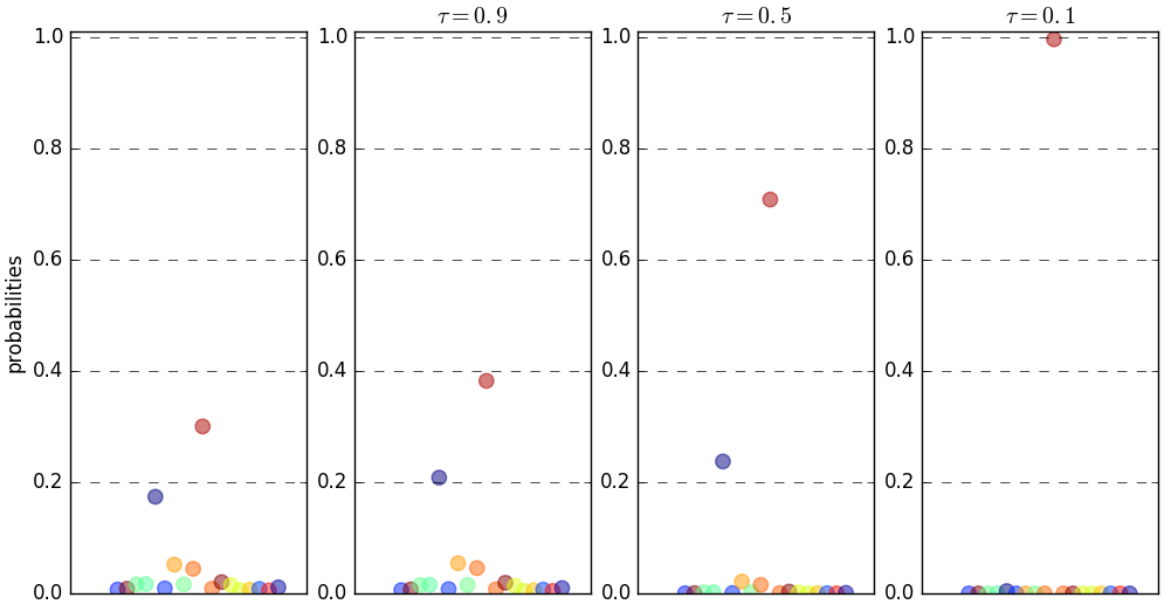

Figure 14: Change in the probabilities depending on the temperature value

6.4 Training Execution

All the models were trained using Pytorch[11], an open source DL framework for Python developed by Facebook's AI research group, and the training process is carried out using a GPU with 12GB.

All the architectures are trained end-to-end, except LDA-Conc, where the LDA model was pre-trained on the descriptions of the TRAIN set, using the *scikit-learn* library (Pedregosa et al., 2011) using the variational Bayes algorithm.

The minimum number of training epochs is 5, and the neural networks continue training until the loss on the DEV set does not decrease in two consecutive epochs, an early stopping rule used to avoid overfitting. All training processes are summarized in Figure 15. Once the training process finished, the best model for each architecture was determined by means of the loss on the DEV set (Table 12).

	Epoch	Training time	Loss	
			TRAIN	DEV
ProductID	12	∼ 4 h/epoch	3.51	4.00
LSTM-LSTM	20	∼ 3 h/epoch	3.51	3.81
LSTM-Att	10	∼ 20 h/epoch	3.29	3.78
LSTM-Conc	18	∼ 3 h/epoch	3.43	3.84
LDA-Conc	20	∼ 4 h/epoch	3.48	3.85

Table 12: Best model for each architecture

[11] https://pytorch.org/

Overall, LSTM-Att needed less time than the other models to train and obtained better loss scores on both TRAIN and DEV datasets. However, processing time of each epoch was much larger in this architecture, as it needs to compute the context vector at each decoding step.

The other models took similar processing time to reach comparable scores on the development set, to the exception of the ProductID baseline, which obtained the worst scores.

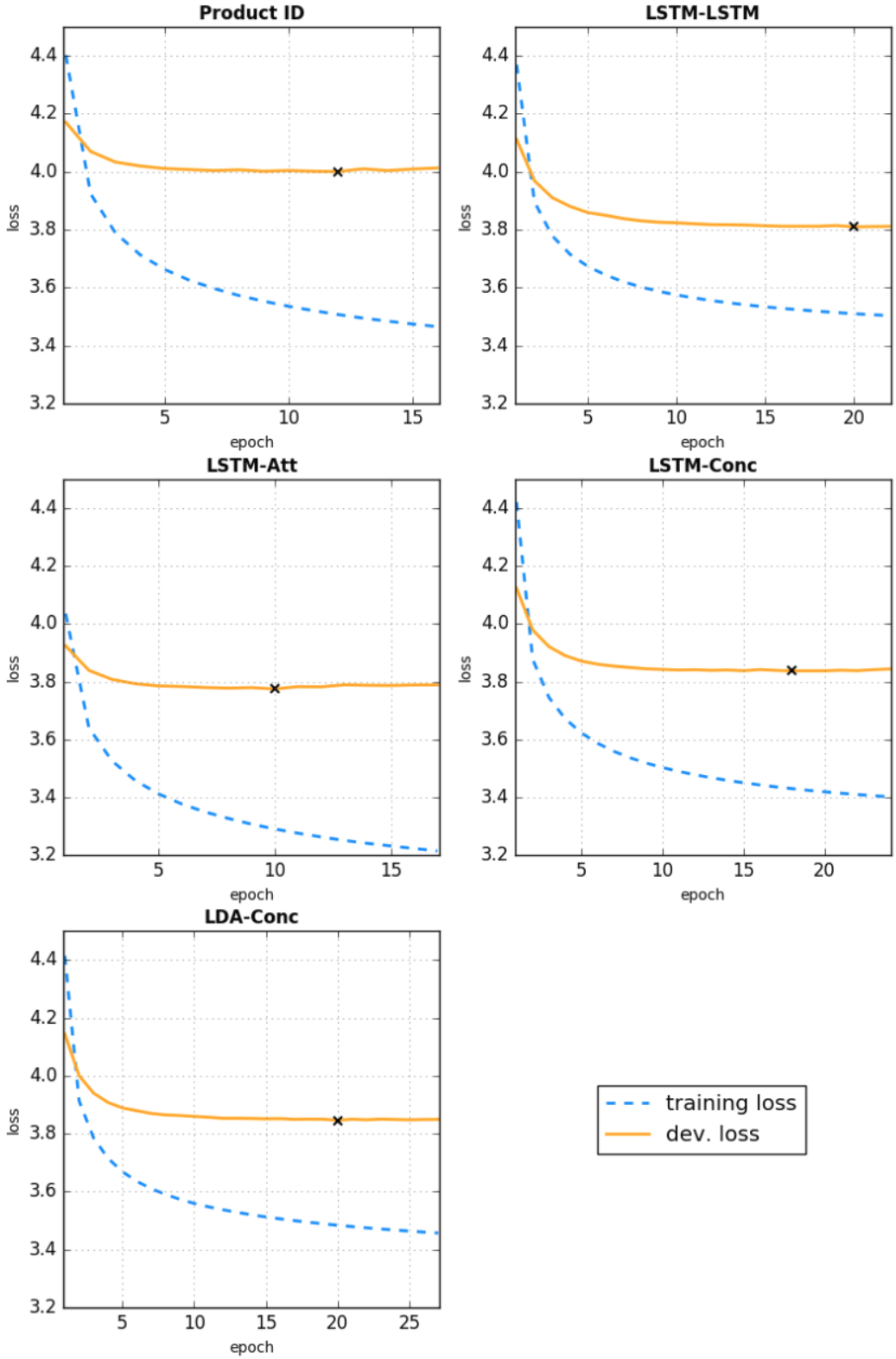

Figure 15: Training process

7 Results and Discussion

The systems were tested by generating new reviews with the ratings and the product descriptions of the test set. Although the maximum sequence length of the decoder was set to 135 tokens, the generation process ends if the special token <EOS> is predicted. Before introducing the description to the model, it is split using the BPE model trained with on the training descriptions, or, in the case of LDA-Conc, is directly encoded using the pre-trained LDA model. Once the reviews were generated, the evaluation methods described in Section 3 were applied.

7.1 Results

For the results reported here, the temperature parameter was set to $\tau = 0.3$, as it provided an optimal balance between text coherence and output variability in our experiments on the development set (Figure 16). Examples in Table 13 show how the temperature value affects the generation process.

Description: *"The classic 1959 Christie tale is revisited in this utterly charming and intriguing murder mystery in which two teachers find the body of..."*

ML: "i have read all of the books in this series and they are all great . i have read all of them and have enjoyed them all ."

$\tau = \mathbf{0.3}$: "i have read all of agatha christie 's books and i have to say that this one is my favorite . i love the characters and the mystery . the story line is well written and the characters are well developed . i really enjoyed this book ."

RS: "to say that suspense got a " feel good " tome eh is not a great puzzle for me . not an avid reader , requiring lots of interesting ideas about this tale . i am a mystery reader and especially of agatha christie 's favor . the book really intrigues me , and i read a lot of books in the newspaper . i recomend this book read to a one-dimensional class and up there with the new rules ! !"

Table 13: Examples of reviews generated with different temperature values

Table 14 presents the results obtained by each architectural variant on the test set. Statistical significance tests were computed via bootstrap resampling (Koehn, 2004). The first noticeable result is that variants differ in terms of their exploitation of the unstructured information provided by the descriptions. Thus, the LSTM-LSTM architecture does not benefit from it, as it obtained similar or lower results than the ProductID baseline across the board. The other three variants, on the contrary, obtained significantly better results overall, demonstrating the benefits of including this type of unstructured information, provided the architecture enables its proper propagation to the decoder level.

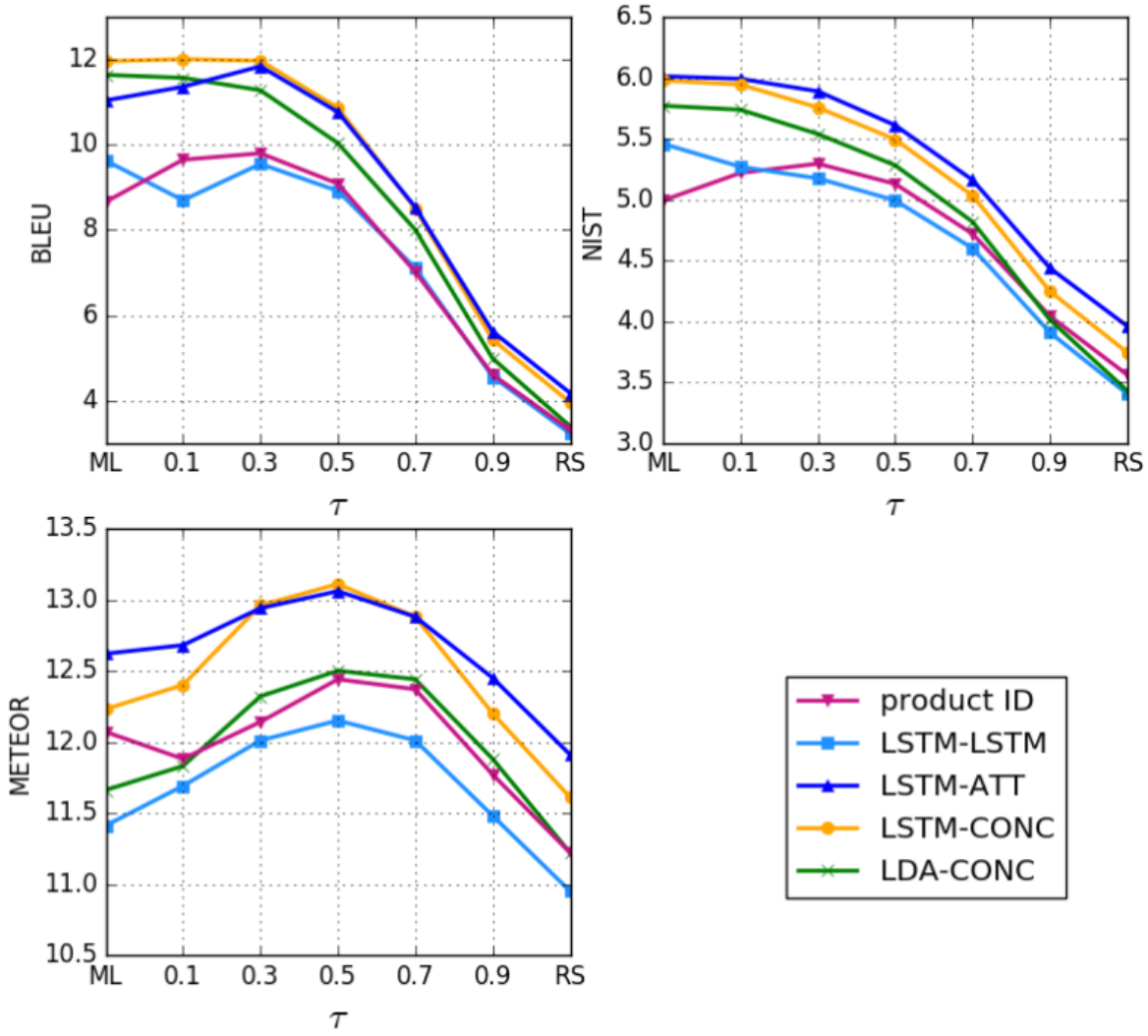

Figure 16: Results on DEV with different temperature values

	METEOR	BLEU-4	BLEU-1	NIST-4	NIST-1
ProductID	12.15	9.90	64.58	5.3191	4.3957
LSTM-LSTM	12.00	9.56	61.74	5.1749	4.3062
LSTM-Att	**12.99***	11.87*	**65.3***†	**5.8889***†	**4.8072***†
LSTM-Conc	12.97*	**12.11***†	64.71	5.7805*	4.6937*
LDA-Conc	12.38*	11.00*	61.69	5.4910*	4.4977*

*: Significantly better than the baseline ($p < 0.05$)
†: Significantly better than all other systems ($p < 0.05$)

Table 14: Results on the test set

Description: *"Mindy Starns Clark is the author of many books (more than 450,000 copies sold), which include A Pocket Guide to Amish Life, Shadows of Lancaster County, Whispers of the Bayou, and The Amish Midwife. In addition, Mindy is a popular inspirational speaker and playwright."*

ProductID: "i read this book when i was a teenager and i read it at least <num> times . i was so excited to find out what happened to the characters and the story was so good that i could n't put it down . i have read it many times and i am sure i will be reading it again . i will be reading it again ."

LSTM-Att: "i really enjoyed this book . i have read all of the amish books and this one was a great one . i really enjoyed the way the author made the characters come alive . the author did a great job of keeping the reader interested and wanting to get to the next book . i would recommend this book to anyone who likes amish christian fiction ."

Table 15: Examples of reviews generated by ProductID and LSTM-Att

The impact of description information on the generated output is visible in the example of Table 15. While the review generated by ProductID is very generic, the information from the description is employed by LSTM-Att to create a more specific review, with expresions like "all of the amish books" or "amish christian fiction".

Different results are obtained by the variants that do exploit the contextual information provided by the descriptions, depending on the considered metric. When considering BLEU-4, LSTM-Conc was the optimal variant, with better results than those obtained with LSTM-Att. On the rest of the metrics however, LSTM-Att outperformed all other architectures.

Table 16 shows the results in terms of perplexity and statistics of the generated reviews. Results in terms of perplexity indicate opposite results to those obtained with the other metrics, in terms of the relative ranking of the models. LSTM-LSTM obtained the lowest results, which indicates that the reviews generated by this variant are more in line with the language model trained on the reviews of the TRAIN set. This can be attributed to

	PPL	#voc	avgLength
ProductID	5.37	1245	38.49
LSTM-LSTM	4.94	1991	40.36
LSTM-Att	5.73	3874	40.71
LSTM-Conc	5.34	3416	41.62
LDA-Conc	5.19	2699	41.78

Table 16: Perplexity and statistics of generated reviews. *#voc* indicates the number of different words generated by the models and *avgLength* the average length of the generated reviews.

Description: *"As many as one in 10 women suffer from Polycystic Ovary Syndrome [PCOS] and Colette Harris explains how with the right nutritional approach sufferers can lose weight, improve their skins, overcome exhaustion, depression and mood swings."*

r = 1 : "i was very disappointed with this book . it was very poorly written and the author did n't seem to understand the information . i would not recommend this book to anyone ."

r = 5 : "i have been using this book for a couple of years and it has helped me to understand my body and how to handle my life . i am very happy with the results ."

Table 17: Examples of reviews generated by LSTM-Att with different ratings

the fact that variants such as LSTM-LSTM generate consistently simple reviews, without veering towards the more variable output produced by the other models that exploit the available contextual information. This is confirmed by the number of different words used to generate reviews, where LSTM-Att uses three times more vocabulary than the baseline, as shown in Table 16. The low perplexity obtained overall by all variants also indicates that the use of contextual information provides slightly more unexpected output, as far as the language model is concerned, but keeps the productions within the range of expected content.

As expected, given the use of rating information in the input, the ability to generate reviews that reflect the sentiment expressed by user ratings was not affected by the use of contextual information. Table 17 shows examples of two opposite reviews on the same description to illustrate this point.

To confirm this intuition, we performed sentiment analysis on the generated reviews using the NLTK toolkit (Bird et al., 2009). After training a VADER model (Hutto and Gilbert, 2014), a rule-based method that proved efficient for sentiment analysis precisely in the Amazon product domain, the reviews were analysed and the system provided a value in the range $[-1, 1]$, indicating a sentiment that goes from very negative to very positive. Boxplots in Figure 17 show the correlation between the input rating and the output of the sentiment analyser. Results are very similar in all the architectures, and the lower the rating of a review, the more difficult it was for the sentiment analyser to detect its sentiment. This latter fact is not due to the systems having issues generating negative reviews, since similar results were obtained analysing the sentiment of real reviews in the corpus. Interestingly, negative reviews generated by LSTM-Att were better classified as negative by the sentiment analyser, even better than human-written reviews.

The results on the complete test set might conflate more fine-grained distinctions regarding the output produced by architectural variants, as simple reviews might be produced similarly by the different models, for instance. Table 18 presents the results on the partitioned test set, which provides information to analyse this issue.

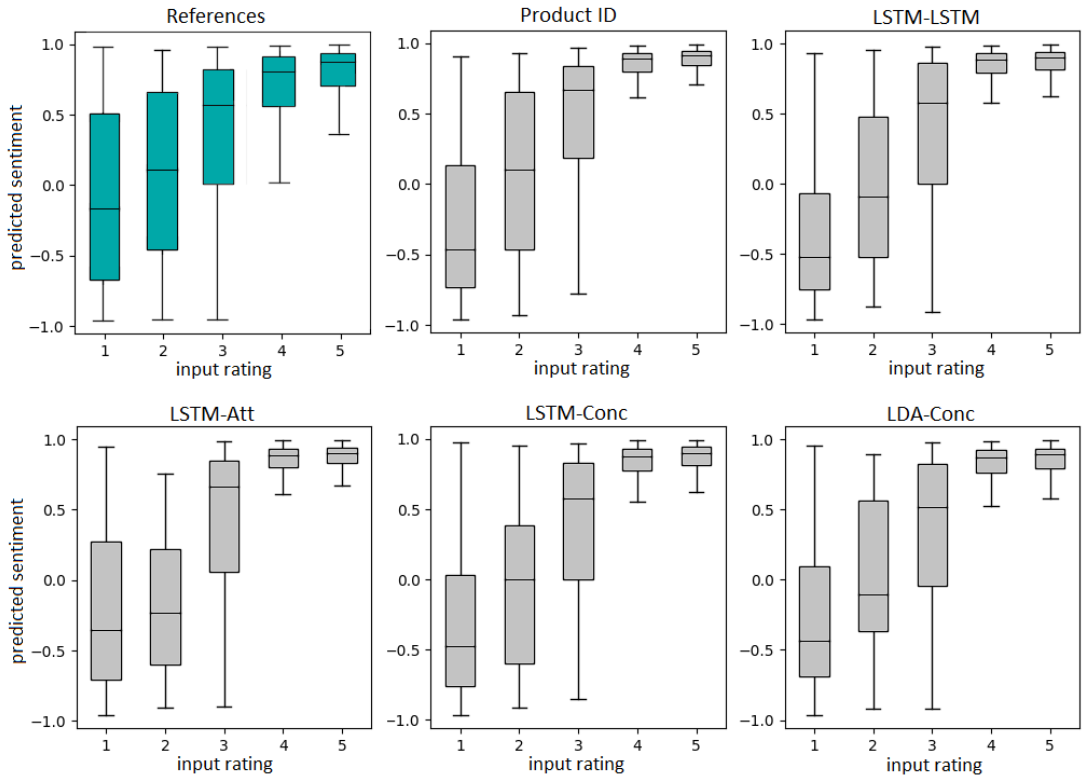

Figure 17: Correlation between ratings and sentiment analysis results

On this test set partition, and except for perplexity measures, all variants see their scores drop significantly as the perplexity on the test reviews increases. The different models are impacted by the increased complexity of the test set partition and the relative hierarchy between them remains similar overall on all three subsets. However, it seems that LSTM-Att is more capable to generate complex reviews, something that we had already observed when analysing the vocabulary and the length of its reviews. This is confirmed by the results on the TEST-HIGH-PERPEXITY set, where LSTM-Att is significantly better than all the other systems in terms of METEOR and NIST-4, which was not the case on the TEST-LOW-PERPLEXITY set. The fact that perplexity scores remain relatively constant for all variants across this partition reinforces our previous interpretation of the results for this metric, indicating that the generated reviews by the different variants closely follow the sequential patterns of the training corpus.

7.2 Discussion

Overall, the LSTM variant with attention obtained the best results on most measures, significantly outperforming the other models, in particular on the NIST metric. Given that this metric favours informational content, these results tend to indicate that the use of an attention mechanism allows for the better exploitation of unstructured contextual

TEST-LOW-PERPLEXITY	PPL	METEOR	BLEU-4	BLEU-1	NIST-4	NIST-1
ProductID	5.38	13.14	9.44	61.86	4.7790	3.9832
LSTM-LSTM	**4.9**	13.15	9.98*	59.56	4.6892	3.9187
LSTM-Att	5.18	**14.16***	13.21*	**62.97*†**	**5.4010***	**4.4047*†**
LSTM-Conc	4.95	14.14*	**13.39***	62.26	5.3684*	4.3461*
LDA-Conc	4.99	13.46*	11.96*	59.82	5.1326*	4.2003*

TEST-MEDIUM-PERPLEXITY	PPL	METEOR	BLEU-4	BLEU-1	NIST-4	NIST-1
ProductID	5.39	10.9	7.39	57.31	4.6786	3.9749
LSTM-LSTM	**4.99**	10.69	6.92	54.7	4.504	3.8544
LSTM-Att	6.05	**11.67***	8.21*	**57.48**	**5.027*†**	**4.2495*†**
LSTM-Conc	5.54	11.62*	**8.48*†**	57.06	4.9314*	4.1533*
LDA-Conc	5.44	10.95	8.03*	55.42	4.8045*	4.047*

TEST-HIGH-PERPLEXITY	PPL	METEOR	BLEU-4	BLEU-1	NIST-4	NIST-1
ProductID	5.41	9.27	5.05	53.56	4.4732	3.9143
LSTM-LSTM	**4.99**	9.23	4.8	51.38	4.2759	3.7591
LSTM-Att	5.92	**10.06*†**	5.61*	**54.46*†**	**4.7525*†**	**4.1368*†**
LSTM-Conc	5.61	9.91*	**5.86***	53.43	4.646*	4.0345*
LDA-Conc	5.53	9.47*	5.85*	52.59	4.5413*	3.941*

*: Significantly better than the baseline ($p < 0.05$)
†: Significantly better than all other systems ($p < 0.05$)

Table 18: Results on the perplexity-split test sets

information in the experiments reported here. Nevertheless, the scores for this system are closely followed by those obtained with LSTM-Conc, a simpler approach that outperforms LSTM-Att in terms of BLEU-4.

The similarity between the results of LSTM-Att and LSTM-Conc are rather surprising. While the former uses a complex attention mechanism, which gives excellent results in machine translation, the latter uses a simpler concatenation mechanism to propagate the encoded information. This similarity may be due to the fact that, as has been verified through a qualitative analysis of the generated reviews, there is no alignment between the information of the description and the review, contrary to the type of relation between the input and the output of machine translation systems. In our task, the context vector is very similar for all generation steps. Therefore, although it encodes the input better than the LSTM and gives better results in terms of the informational content, the main advantage of the attention mechanism, i.e. focusing attention on different points of the input throughout the process, is less useful for the generation task.

The different results obtained on the various test sets reveals the importance of the evaluation data, since the BLEU-4 score of the same system falls by more than 7 points from one set to another. For instance, LSTM-ATT goes from 13.21% to 5.61% depending on the evaluation being performed on TEST-LOW-PERPLEXITY OR TEST-HIGH PERPLEXITY.

Beyond comparisons with the baseline, the lack of a gold-standard metric to evaluate

NLG systems makes it difficult to directly compare our results to those obtained in other studies. Note though that the BLEU results we obtained are in line with those obtained in reference NLG papers on review generation (Dong et al., 2017; Zang and Wan, 2017), which are typically lower than the scores obtained in neural machine translation, where there is stronger conditioning on the source language information to produce a more controlled output.

8 Conclusions and Future Work

In this work, we presented a novel task for natural language generation, based on the exploitation of unstructured contextual information. The aim of the task is to provide a test bench to compare different architectures, as it enables an evaluation of a system's capability to generate coherent text based on unseen and unstructured information.

We described a variant of the Amazon Product Review corpus for the task, with product descriptions used as input for the generation of reviews. The corpus was prepared to enable fair evaluations of different architectures, notably including different partitions of the test set in terms of review complexity and enforcing an adequate number of review references per description.

To evaluate the discriminative power of the proposed task and measure the impact of different NLG architectural variants, we built different deep learning generation models and compared their results on different metrics. Although the nature of the NLG task and the currently available metrics make it rather difficult to measure the impact of model variants, the proposed task enabled us to assess significant differences between the tested architectures, showing that the model doted with an attention mechanism was the best variant among the tested ones overall.

However, our results also indicate that a much simpler architecture using a RNN with concatenated encoder information is able to take advantage of the unstructured information and achieve comparable results. Although the results are slightly worse than those obtained with attention, it is a much faster architecture both to implement and to train. Depending on the needs of the task and the available computational resources, it can end up being a more useful architecture than the one endowed with an attention mechanism.

Although we showed that several of the variants were able to exploit the provided unstructured contextual information, these results are preliminary and future work will address shortcomings of the current task, such as the inclusion of descriptions and reviews that bear no relation to each other. We also aim to explore the performance on this task on other novel architectures, such as hierarchical models, variational autoencoders or generative adversarial networks. It would also be interesting to try an architecture that encodes the description in a similar way to the one used in the attentional variant, but without recomputing at each generation step, as we demonstrated that this does not improve the results while lengthening considerably the training process. Finally, we aim to improve the evaluation of this task by means of human evaluation and extend the use of unstructured input to other data-to-text tasks, to further expand the available options for the comparison of NLG architectures.

References

R. Arun, V. Suresh, C. E. Veni Madhavan, and M. Narasimha Murty. On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 1, pages 391–402, Hyderabad, India, 2010. Springer. URL http://doi.org/10.1007/978-3-642-13657-3_43.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015. URL http://arxiv.org/pdf/1409.0473.pdf.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2322.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, Dennis Morello, and Fabiano Tarlao. "Best Dinner Ever!!!": Automatic generation of restaurant reviews with LSTM-RNN. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 721–724, Omaha, NE, USA, 2016. IEEE Computer Society. URL http://doi.org/10.1109/WI.2016.0130.

Anja Belz and Albert Gatt. Intrinsic vs. extrinsic evaluation measures for referring expression generation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: HLT, Short Papers*, pages 197–200, Columbus, OH, USA, 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P08-2050.

Anja Belz and Ehud Reiter. Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, 2006. The Association for Computer Linguistics. URL http://www.aclweb.org/anthology/E06-1040.

Y Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *1993 IEEE International Conference on Neural Networks*, volume 3, pages 1183 – 1188, San Francisco, CA, USA, 1993. URL http://doi.org/10.1109/ICNN.1993.298725.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. URL http://doi.org/10.1109/72.279181.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. URL http://doi.org/10.1109/TPAMI.2013.50.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 978-0-596-51649-9.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003. URL http://www.jmlr.org/papers/v3/blei03a.html.

Giuseppe Carenini and Johanna D. Moore. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952, 2006. URL http://doi.org/10.1016/j.artint.2006.05.003.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, CA, USA, 2016. The Association for Computational Linguistics. URL http://aclweb.org/anthology/N/N16/N16-1012.pdf.

Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, pages 57:1–57:2, Tokyo, Japan, 2018. URL http://arxiv.org/pdf/1707.01561.pdf.

Michael J. Denkowski and Alon Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2107.

Romain Deveaud, Eric SanJuan-Ibekwe, and Patrice Bellot. Accurate and effective latent concept modeling for ad hoc information retrieval. *Document Numérique*, 17(1):61–84, 2014. URL http://doi.org/10.3166/dn.17.1.61-84.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Diego, CA, USA, 2002. Morgan Kaufmann Publishers. URL http://www.mt-archive.info/HLT-2002-Doddington.pdf.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1,*

Long Papers, pages 623–632. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/E17-1059.

Ondřej Dušek and Filip Jurcicek. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 2: Short Papers*, pages 45–51, Berlin, Germany, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-2008.

Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. Referenceless quality estimation for natural language generation. *CoRR*, abs/1708.01759, 2017. URL http://arxiv.org/pdf/1708.01759.pdf.

Adil El Ghali. Use of description logic and SDRT in an nlg system. In *Proceedings of the International Natural Language Generation Conference*, pages 179–184, Harriman, NY, USA, 2002. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W02-2124.

Michael Elhadad and Jacques Robin. An overview of SURGE: a reusable comprehensive syntactic realization component. In *Proceedings of the Eighth International Natural Language Generation Workshop, Posters and Demonstrations*, Brighton, England, 1996. URL http://www.aclweb.org/anthology/W96-0501.

Desmond Elliott and Arjen P. de Vries. Describing images using inferred visual dependency representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing*, pages 42–52, Beijing, China, 2015. URL http://aclweb.org/anthology/P/P15/P15-1005.pdf.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. URL http://doi.org/10.1207/s15516709cog1402_1.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark, 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/W17-4912.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, CA, USA, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1087.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994.

Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018. URL http://doi.org/10.1613/jair.5477.

Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert: Intelligent Systems and Their Applications*, 9(2):45–53, 1994. URL http:/doi.org/10.1109/64.294135.

Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57(1):345–420, 2016. URL http://arxiv.org/pdf/1510.00726.pdf.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613.

Raghav Goyal, Marc Dymetman, and Eric Gaussier. Natural language generation through character-based rnns with finite-state prior knowledge. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1083–1092, Osaka, Japan, 2016. The COLING 2016 Organizing Committee. URL http://www.aclweb.org/anthology/C16-1103.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/pdf/1308.0850.pdf.

T. L. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, volume 101, pages 5228–5235, 2004. URL http://doi.org/10.1073/pnas.0307752101.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Long Papers*, volume 1, Berlin, Germany, 2016. URL http://www.aclweb.org/anthology/P16-1014.

Kenneth Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2123.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/pdf/1207.0580.pdf.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/pdf/1503.02531.pdf.

Matthew D. Hoffman, David M. Blei, and Francis Bach. Online learning for latent dirichlet allocation. *Neural Information Processing Systems*, 23:856–864, 2010. URL http://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1587–1596, Sydney, Australia, 2017. PMLR. URL http://proceedings.mlr.press/v70/hu17e.html.

Clayton J. Hutto and Eric Gilbert. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media*, Ann Arbor, MI, USA, 2014. The AAAI Press. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109.

A.G. Ivakhnenko and V.G. Lapa. *Cybernetic Predicting Devices*. Jprs report. CCM Information Corporation, 1965.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, Boston, MA, USA, 2015. IEEE Computer Society. URL http://doi.org/10.1109/CVPR.2015.7298932.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. URL http://arxiv.org/pdf/1506.02078.pdf.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, TX, USA, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D16-1032.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, pages 1–13, 2015. URL http://arxiv.org/pdf/1412.6980.pdf.

Sheldon Klein. *Automatic Novel Writing: A Status Report*. Technical report. University of Wisconsin, Madison, WI, USA, 1973.

Kevin Knight and Vasileios Hatzivassiloglou. Two-level, many-path generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 252–260, Cambridge, MA, USA, 1995. Morgan Kaufmann Publishers / ACL. URL http://www.aclweb.org/anthology/P95-1034.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages

388–395, Barcelona, Spain, 2004. ACL. URL http://www.aclweb.org/anthology/W04-3250.

Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992. URL http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf.

Irene Langkilde. Forest-based statistical sentence generation. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 170–177, Seattle, WA, USA, 2000. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/A00-2023.

Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://www.cs.cmu.edu/~alavie/papers/BanerjeeLavie2005-final.pdf.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, London, UK, 1999. Springer-Verlag. URL http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015. URL http://www.nature.com/articles/nature14539.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. 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, pages 1106–1115, Beijing, China, 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1107.

Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. Capturing meaning in product reviews with character-level generative text models. *CoRR*, abs/1511.03683, 2015. URL http://arxiv.org/abs/1511.03683.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. 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, pages 11–19, Beijing, China, 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1002.

William C. Mann. An overview of the PENMAN text generation system. In *Proceedings of the third national conference on artificial intelligence*, pages

261–265, WA, USA, 1983. URL http://pdfs.semanticscholar.org/908b/0130d8563afb96e29e743a8f638fb3312ba9.pdf.

Jonathan May and Jay Priyadarshi. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 536–545, Vancouver, Canada, 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S17-2090.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, Santiago, Chile, 2015. ACM. URL http://arxiv.org/pdf/1506.04757.pdf.

David D. McDonald. Issues in the choice of a source for natural language generation. *Computational Linguistics*, 19(1):191–197, 1993. URL http://www.aclweb.org/anthology/J93-1009.

Susan W. McRoy, Songsak Channarukul, and Syed S. Ali. YAG: A template-based generator for real-time systems. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, pages 264–267, Dublin, Ireland, 2000. URL http://www.aclweb.org/anthology/W00-1437.

Susan W. Mcroy, Songsak Channarukul, and Syed S. Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.459.3460&rep=rep1&type=pdf.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1086.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048. ISCA, 2010. URL http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K16-1028.

Jianmo Ni, Zachary C. Lipton, Sharad Vikram, and Julian McAuley. Estimating reactions and recommending products with generative models of reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 1, pages 783–791, Taipei, Taiwan, 2017. Asian Federation of Natural Language Processing. URL http://aclweb.org/anthology/I17-1079.

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. Crowd-sourcing NLG data: Pictures elicit better data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 265–273, Edinburgh, UK, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-6644.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark, 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/D17-1238.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA, 2002. URL http://www.aclweb.org/anthology/P02-1040.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012. URL http://arxiv.org/pdf/1211.5063v1.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

Paul Piwek. A flexible pragmatics-driven language generator for animated agents. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 151–154, Budapest, Hungary, 2003. URL http://www.aclweb.org/anthology/E03-1062.

François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artif. Intell.*, 173(7-8):789–816, 2009. URL http://doi.org/10.1016/j.artint.2008.12.002.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-6603.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444, 2017. URL http://arxiv.org/pdf/1704.01444.pdf.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Joseph Pal, and Aaron C. Courville. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 241–251, Vancouver, Canada, 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/W17-2629.

Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009. URL http://www.aclweb.org/anthology/J09-4008.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-02451-X.

Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2152–2161, Lille, France, 2015. PMLR. URL http://proceedings.mlr.press/v37/romera-paredes15.pdf.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958. URL http://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL http://doi.org/10.1038/323533a0.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117, 2015. URL http://doi.org/10.1016/j.neunet.2014.09.003.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, Copenhagen, Denmark, 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/D17-1066.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1162.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562, Melbourne, Australia, 2015. ACM. URL http://arxiv.org/pdf/1507.02221.pdf.

Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1017–1024, Bellevue, WA, USA, 2011. Omnipress. URL http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112, Montreal, Canada, 2014. MIT Press. URL http://arxiv.org/pdf/1409.3215.pdf.

Jian Tang, Yifan Yang, Samuel Carton, Ming Zhang, and Qiaozhu Mei. Context-aware natural language generation with recurrent neural networks. *CoRR*, abs/1611.09900, 2016. URL http://arxiv.org/pdf/1611.09900.pdf.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. RUBER: an unsupervised method for automatic evaluation of open-domain dialog systems. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, New Orelasn, LA, USA, 2018. AAAI Press. URL http://lili-mou.github.io/paper/2018-AAAI-RUBER.pdf.

Van-Khanh Tran, Le-Minh Nguyen, and Satoshi Tojo. Neural-based natural language generation in dialogue using RNN encoder-decoder with semantic aggregation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 231–240, Saarbrücken, Germany, 2017. Association for Computational Linguistics. URL http://aclanthology.info/papers/W17-5528/w17-5528.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99, 2017. URL http://aclweb.org/anthology/Q17-1007.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*. IEEE Computer Society, 2015a. URL http://doi.org/10.1109/ICCV.2015.515.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1494–1504, Denver, CO, USA, 2015b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1173.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D15-1199`.

Terry Winograd. *Understanding Natural Language*. Academic Press, Inc., Orlando, FL, USA, 1972. ISBN 9780127597508.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/pdf/1609.08144.pdf`.

Han Xiao and Thomas Stibor. Efficient collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of 2nd Asian Conference on Machine Learning*, volume 13, pages 63–78, Tokyo, Japan, 08–10 Nov 2010. PMLR. URL `http://proceedings.mlr.press/v13/xiao10a/xiao10a.pdf`.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, volume 37, pages 2048–2057, Lille, France, 2015. JMLR. URL `http://proceedings.mlr.press/v37/xuc15.pdf`.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593. IEEE Computer Society, 2016. URL `http://doi.org/10.1109/CVPR.2016.496`.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2852–2858, San Francisco, CA, USA, 2017. AAAI Press. URL `http://arxiv.org/pdf/1609.05473.pdf`.

Hongyu Zang and Xiaojun Wan. Towards automatic generation of product reviews from aspect-sentiment scores. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 168–177, Santiago de Compostela, Spain, 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W17-3526`.