Ordering Problems in Natural Language Generation

Farhood Farahnak

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Computer Science) at

Concordia University

Montréal, Québec, Canada

April 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Farhood Farahnak**

Entitled: **Ordering Problems in Natural Language Generation**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Name of the Chair

______________________ External Examiner
Dr. Stan Matwin

______________________ Examiner
Dr. Tristan Glatard

______________________ Examiner
Dr. Olga Ormandjieva

______________________ Examiner
Dr. Nizar Bouguila

______________________ Supervisor
Dr. Leila Kosseim

Approved by ______________________
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

__________ 2023

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Ordering Problems in Natural Language Generation

Farhood Farahnak, Ph.D.

Concordia University, 2023

Language is the main mechanism used in human communication. For decades, researchers have tried to build computers capable of communicating with humans in natural languages. Such research endeavors include building systems to understand human language (a.k.a Natural Language Understanding or NLU) and generate a response in human language (a.k.a Natural Language Generation or NLG). The goal of NLG is to deliver information by producing natural texts in human languages. Text generation typically consists of several steps in a pipeline, from determining the content to communicate to producing the actual words, and hence requires different techniques at each step. In this thesis, we focus on ordering problems in NLG and address two tasks that require ordering; namely, sentence ordering and surface realization.

In sentence ordering, models need to capture the relations between sentences and then based on these relations, find the most coherent order of sentences. Our proposed approach is based on pointer networks where at each step, the model chooses the sentence that should appear next in the text. We show that using a conditional sentence representation that captures the sentence meaning based on its position and previously selected sentences in the text can improve the state-of-the-art on standard datasets.

For surface realization, we show that a pointer network is insufficient to improve state-of-the-art performance. Therefore, we propose the use of language models for surface realization by mapping the surface realization task (a graph-to-text) to a text-to-text problem. Our experiments show that pre-trained language models can easily learn the task of surface realization and achieve competitive performances on standard datasets. To further improve the performance of surface realization, we then propose to pre-train a language model on synthetic data and then fine-tune it on manually labeled data in order to increase the number of training data for surface realization. Our experiments indicate that this approach improves the state-of-the-art by more than 10% BLEU score on standard datasets.

Acknowledgments

I would like to take this opportunity to deeply thank my supervisor, Prof. Leila Kosseim, for her insightful advice, continuous support, and patience during my Ph.D. study. She is a role model in my academic career, and I hope to pay forward what she has done for me.

I would also like to thank my thesis committee, Dr. Olga Ormandjieva, Dr. Tristan Glatard, Dr. Nizar Bouguila, and my external committee Dr. Stan Matwin for their constructive feedback and continued support.

I am always indebted to my family, mother, father, brother, and dearest sister, Farzaneh, whose unconditional support and inspiration have always helped me my whole life.

My deepest gratitude goes to my partner, Laya. From the beginning of my Ph.D. journey, she stood by my side in all the ups and downs. From emotional support to scientific advice, she was my one and only one. I cannot imagine achieving this goal without her support.

In the end, I would like to express my sincere respect for Iranian women fighting for their rights. Woman, Life, Freedom.

Contribution of Authors

The main chapters in this dissertation (Chapters 3, 4, 5, and 6) are based on papers already published. Below is a description of our individual contribution.

- Chapter 3 on using Pointer Networks for Surface Realization has been published as Farahnak, Rafiee, Kosseim, and Fevens (2019): Farahnak, F., Rafiee, L., Kosseim, L., and Fevens, T. (2019). The Concordia NLG Surface Realizer at SRST 2019. In Proceedings of the 2nd workshop on multilingual surface realisation (MSR 2019) (pp. 63–67).

  Contributions of Authors: Farhood Farahnak led the project, writing the code for the pointer network and the encoder, running the experiments and writing the paper. Laya Rafiee contributed to running a few experiments and co-writing the paper. Leila Kosseim and Thomas Fevens contributed to editing the paper and provided general guidance and technical direction.

- Chapter 4 on using pre-trained Language Models for Surface Realization has been published as Farahnak, Rafiee, Kosseim, and Fevens (2020): Farahnak, F., Rafiee, L., Kosseim, L., and Fevens, T. (2020, December). Surface realization using pretrained language models. In Proceedings of the 3rd workshop on multilingual surface realisation (MSR 2020) (pp. 57–63).

  Contributions of Authors: Farhood Farahnak led the project, writing the code the for neural language model and training it, running the experiments and writing the paper. Laya Rafiee contributed to running additional experiments and co-writing the paper. Leila Kosseim and Thomas Fevens contributed to editing the paper and provided general guidance and technical direction.

- Chapter 5 on Using conditional representation for Sentence Ordering has been published as

Farahnak and Kosseim (2021): Farahnak, F., and Kosseim, L. (2021). Using conditional sentence representation in pointer networks for sentence ordering. 2021 IEEE 15th International Conference on Semantic Computing (ICSC 2021), (pp. 288-295).

Contributions of Authors: Farhood Farahnak led the project, writing the code for the pointer network and the conditional encoder, running the experiments and writing the paper. Leila Kosseim contributed to editing the paper and provided general guidance and technical direction.

- Chapter 6 on pre-training Language Models with synthetic data has been published as Farahnak and Kosseim (2022): Farahnak, F., and Kosseim, L. (2022, December). Pre-training language models for surface realization. In Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022) (pp. 312–318).

  Contributions of Authors: Farhood Farahnak led the project, writing the code for the training and creating the synthetic data, running the experiments and writing the paper. Leila Kosseim contributed to editing the paper and provided general guidance and technical direction.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 1

AMR Abstract Meaning Representation. 5, 8

HMM Hidden Markov Model. 39

LDA Latent Dirichlet Allocation. 15

LM Language Model. 8, 9, 12, 13, 17, 69

MSR Multilingual Surface Realization. 16, 68, 69

NLG Natural Language Generation. 1–6, 9, 10, 18, 19, 54

NLP Natural Language Processing. 1

OWL Web Ontology Language. 8

RDF Resource Description Framework. 8

SOTA state-of-the-art. 12

SR Surface Realization. 17

UD Universal Dependency. xi, 28–31, 33

Chapter 1

Introduction

1.1 Motivation

We (humans) easily communicate with each other using natural languages; however, this is not the case when we communicate with machines and computers. Communicating via human languages makes it easier for humans to use and benefit from computers without requiring to learn any specific skill. This is the case when people communicate with virtual assistants such as Google Assistant or Siri. However, today, the usage of such systems is limited to pre-defined use cases, and extending them to more general and all-purpose use cases, requires improvement in Natural Language Processing (NLP). Work in NLP generaly address two main goals: creating systems to understand human languages or to generate texts in human languages.

Natural Language Generation (NLG) is a subfield of NLP, itself a subfield of Artificial Intelligence (AI). According to Reiter and Dale (2000): "NLG is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information." The input representation can take a variety of forms such as tabular data, images, knowledge graphs, or unstructured text. NLG is the basis for a wide range of applications such as generating weather reports, text summarization, and conversational models.

Depending on their final application and/or their architecture, NLG systems can be classified into four families of approaches. The first type of approach, and maybe the oldest, generates a text by filling in values and information in predefined templates. One example of systems using this approach is weather forecast generator of Glahn (1970). The second type of approach involves the selection of the most appropriate output from a predefined set of responses. Examples of systems that follow this approach include question answering (Yang, Yih, & Meek, 2015) and extractive summarization models (Nallapati, Zhai, & Zhou, 2017). The third type of approach used in NLG involves the production of text from scratch. In this case, the model generates words (or sub-words) one by one. Examples of NLG systems that follow this approach include models for machine translation (Vaswani et al., 2017) and conversational agents (Li et al., 2016). Finally, a fourth family of approaches involves ordering a given set of inputs. This includes word ordering in surface realization and sentence ordering tasks.

The general goal of this thesis is to study the last approach described above: ordering problems in NLG. We consider two sub-tasks in NLG: Sentence Ordering and Surface Realization. Surface Realization generates the surface form of a sentence from an underlying graph representation by ordering the nodes (words) in the graph and inflecting them. Sentence ordering, on the other hand, aims to generate a paragraph from a given set of sentences by arranging these sentences in their most coherent and logical order.

Ordering a set of items, in our case, words and sentences, require the model to capture and infer the relation among these items to be able to find their correct order. In case of sentence ordering, the model not only needs to capture the meaning of a sentence, but also needs to relate sentences to their surrounding sentences and hence learn the general structure of the discourse. Similarly, for surface realization, the model should understand the semantic and morphological information of the words and, at the same time, learn how these words interact with each other to construct a meaningful sentence or phrase and hence learn the general grammar of the language. Proposed solutions to improve the performance of these models can also be beneficial to other NLG tasks that require capturing the relation of the elements in text.

1.2 Problem Statement

In this thesis, we propose to develop novel approaches based on deep learning architectures to improve the state of the art in ordering problems in NLG. We consider two sub-tasks of NLG: Sentence Ordering and Surface Realization. Sentence Ordering is involved in finding the most logical and coherent order of a given set of sentences; while Surface Realization aims to sort words and inflect them to constitute a sentence. In this section, we define each of these two tasks.

1.2.1 Sentence Ordering

A multi-sentence text is not only a concatenation of sentences; it needs to exhibit qualities of a discourse. Jurafsky and Martin (2021) define a discourse as a "collocated, structured, coherent groups of sentences". In a well-written text, sentences have logical and semantic relations (coherence) with each other that follow the same topic to convey information regarding a specific subject. Coherent texts make it easier for the reader to understand the relations between the elements of the text and their communicative goals. For example, Example 1 is a coherent text as it is clear that the second sentence provides a `REASON` for the first sentence, whereas Example 2 is not coherent as there does not seem to be any relation between the two sentences.

Example 1

Jane took a train from Paris to Istanbul. She had to attend a conference.

Example 2

Jane took a train from Paris to Istanbul. The temperature was -21 in Montreal yesterday.

Modeling discourse computationally is useful for many downstream applications such as measuring the quality of a written text (Somasundaran, Burstein, & Chodorow, 2014), text summarization and information extraction where knowing which coherence relation exists between sentences can be used to identify the most useful information (Barzilay, Elhadad, & McKeown, 2001; Bollegala, Okazaki, & Ishizuka, 2005); or even in health care where modeling discourse can help detecting symptoms of schizophrenia or other mental diseases correlated with disordered language (Iter, Yoon,

& Jurafsky, 2018). Modeling text coherence has been well studied in the context of NLG. In particular, researchers have studied how to automatically identify the coherence relation between two spans of a text (Cianflone & Kosseim, 2018), how to measure text coherence (Somasundaran et al., 2014), and how to order sentences automatically (Lapata, 2003).

Sentence ordering, as a sub-task of coherence modeling, aims to arrange a given set of sentences into their most coherent order. Although by reading a bag of sentences in a random order, readers can probably understand the overall gist of the text, they will not be able to properly identify the relations among the elements of the text. This is because a discourse is not a bag of sentences: the order of sentences signals semantic and logical relations. Proper sentence ordering can entail coherence relations between the entities in the text and clarify the communicative goal of each element of the text.

Sentence ordering can be used in several downstream tasks that require multi sentence generation. For example, in multi document text summarization, the model first chooses a subset of sentences that summarize the content of each document, then a sentence ordering module arranges them in their most coherent order (Barzilay et al., 2001). Sentence ordering can also be beneficial to other applications such as concept-to-text generation (Konstas & Lapata, 2012), storytelling (Hu et al., 2020), and essay scoring (Burstein, Tetreault, & Andreyev, 2010).

Several models have been proposed to address the task of sentence ordering using pointer networks (Vinyals, Fortunato, & Jaitly, 2015) (see Section 2.3). These models achieved promising results and improved the state-of-the-art in sentence ordering, however, they do not capture well the local dependencies between sentences. Indeed, sentences are encoded individually regardless of their position. Our thesis extends this line of work and improves their performance by better capturing the relation between sentences.

1.2.2 Surface Realization

The second ordering task we address is surface realization. Traditionally, Reiter and Dale (2000) divided the steps of an NLG pipeline into three main stages[1]:

(1) Document Planning: Determining the content and structure of a document.

(2) Micro Planning: Deciding which words and syntactic structures will be used to communicate the content.

(3) Surface Realization: Mapping the abstract representation into an actual text.

Surface realization is the process of deriving the surface text, usually individual sentences, from a more Abstract Meaning Representation (AMR) of that text. The goal of the surface realizer, at the end of an NLG pipeline, is to generate the actual words (or surface form). For example, given the abstract input form: (`Subject: i, Object: montreal, Verb: visit, Tense: past perfect`), the surface realization should generate: *I have visited Montreal.*[2]

Surface realization typically involves three tasks: syntactic realization, morphological realization, and orthographic realization (Reiter & Dale, 2000). Whereas morphological and orthographic realizations are responsible for word inflections, punctuation, and formatting, syntactic realization tries to identify the proper linear ordering of the input data. Hence, syntactic realization can be addressed as an ordering task.

Unlike many tasks in NLG, the state-of-the-art performance of surface realization models is still below human performance and hence is an active research area. One of the main obstacle in training neural models for surface realization is the insufficient amount of labeled training data. This is why we investigated different ordering approaches to improve the performance of models for the task of surface realization and proposed an approach to overcome the problem of limited label data for this task.

[1]Different NLG systems may combine a subset (or all) of these stages into a single module or even expand these stages into more fine-grained steps depending on the architecture or use cases. In many recent NLG systems, a single module is responsible for more than one stage which makes it hard to distinguish between different stages in these systems. For example, the neural image captioning model of Vinyals, Toshev, Bengio, and Erhan (2015), uses a single LSTM module for text generation where this module is responsible for all the above-mentioned stages.

[2]Or any semantically equivalent sentence.

1.3 Research Objectives

The general goal of our research is to investigate and design neural network models for sentence ordering and surface realization in order to improve the performance of NLG systems. These two tasks are similar in the sense that they are both involved in finding the correct order of some input. Hence, we can formulate them both as an ordering problem. The previously proposed sentence ordering models (e.g. Logeswaran, Lee, and Radev (2018)) were able to achieve promising results but still had room for improvement as they did not address the local dependencies accross sentences. On the other hand, the performance of surface realization models currently stands below the human level. This is mainly due to limited available labeled training data. In this thesis, we propose a model to reduce the gap between human performance and machine performance in surface realization. We will also try to improve the performance of these models without the need to increase the amount of labeled training samples using transfer learning approaches. The specific research questions we address are as follow:

- RQ1. Can we address the problem of surface realization as an ordering problem? (See Chapter 3)
- RQ2. Can we use a standard (text-to-text) language model to address the task of surface realization? (See Chapter 4)
- RQ3. Can we design a sentence representation that captures the coherence relations between sentences? (See Chapter 5)
- RQ4. Can we improve the performance of surface realizer models with less amount of labeled training data than the current state-of-the-art models? (See Chapter 6)

1.4 Overall Methodology

As indicated in Section 1.3, our goal is to improve the state-of-the-art in two NLG tasks: sentence ordering and surface realization using novel neural networks. In both tasks, the goal is to find the

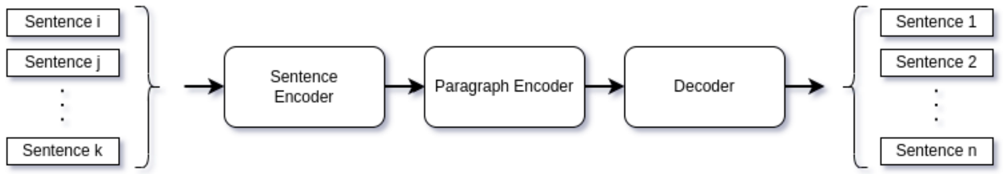


Figure 1.1: General architecture of our proposed Sentence Ordering model

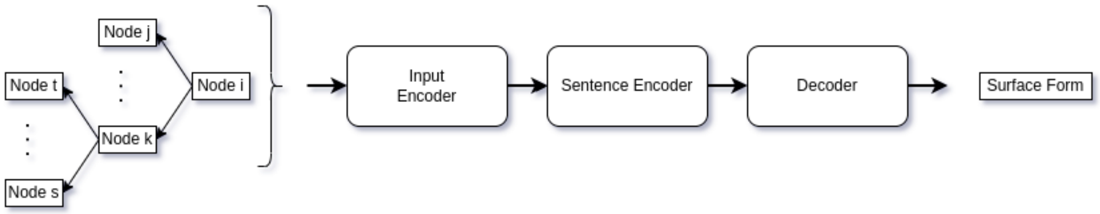


Figure 1.2: General architecture of our proposed Surface Realization model

correct order of inputs and can be formulated as an ordering problem. Let us describe our overall methodology in the following sections.

1.4.1 Sentence Ordering

The main objective of sentence ordering is to arrange sentences in a text in their most logical order in order to maximize coherence. To achieve this, we propose to use an encoder-decoder architecture. The initial set of sentences will be fed to a module to find a representation of each sentence (Sentence Encoder). These representations will then be used by another module to find the relations between sentences (Paragraph Encoder). In the end, the last module (Decoder) will find the most appropriate order of sentences. Figure 1.1 shows an overview of our proposed pipeline for sentence ordering. The training data for the task of sentence ordering needs to have 1) a set of sentences that belong to the same topic as the input and 2) the correct permutation of sentences that preserve the required features which is a coherent text as the target or output. In order to provide these features, we can use well-written texts which are easily available online such as scientific papers, short stories, etc. We shuffle the order of the sentences in the text and train the model to rearrange them in their original correct order.

1.4.2 Surface Realization

We consider two approaches to address surface realization. The first approach is based on pointer networks and transformers; while in the second approach takes advantage of pre-trained language models. Here, we briefly explain these two approaches.

The first proposed approach for surface realization is shown in Figure 1.2. As shown in the figure, the overall architecture is similar to that of the sentence ordering model (see Figure 1.1) as it is also based on an encoder-decoder architecture. However, the model for the surface realization works with more abstract objects as the input is an AMR in the form of a graph. Also, the decoder is responsible for two tasks: 1) ordering, i.e. finding the correct order of tokens, 2) inflection, i.e. generating the surface form of each token. Initially, the model encodes each node and its features (Input Encoder), then another module captures the idea or general information of the sentence or phrase that requires to be generated (Sentence Encoder) and finally a Decoder module generates the sentence or phrase.

In the second approach, we propose to use a pre-trained encoder-decoder Language Model (LM) for surface realization. Encoder-decoder language models are mainly designed for text-to-text problems. These models are trained to reconstruct a corrupted input text (denoising auto-encoder) in a self-supervised paradigm. During training, the LM learns the general knowledge of the language and can generate text from its input. However, these models are not designed to deal with graphs as their input. In order to use encoder-decoder LMs for surface realization (a data-to-text task) we need to represent the input graph in a linear text form (e.g. $Node_i$ `relation` $Node_k$ `Features`) and use the surface form of the sentence as the target to train the LM. In this approach, the LM encoder is responsible for encoding both the nodes and the sentences while the LM decoder is in charge of all decoding steps (i.e. ordering and inflection).

Several types of AMR could be used to represent the input. Knowledge representation schemes such as Web Ontology Language (OWL) (Bechhofer, 2009) or Resource Description Framework (RDF) (Pan, 2009) could be used. However, in order to provide a more general solution and compare our system with other work, following previous work (Mille, Belz, Bohnet, Pitler, & Wanner, 2018), we have used syntactic parse trees as the input graph.

For training data, we have used available datasets that contain parse tree information such as the Universal Dependency Treebank (Zeman et al., 2018). However, since the number of labeled data is limited, we also used unlabelled texts, such as wiki-text (Merity, Xiong, Bradbury, & Socher, 2017), and used a dependency parser (Qi, Zhang, Zhang, Bolton, & Manning, 2020) to create syntactic data to pre-train our models and further improve their performance.

1.5 Contributions

The main contributions of this thesis include:

(1) We proposed an approach to Surface Realization that uses transformers and pointer networks to generate coherent texts from an abstract representation.

    In this approach, the transformer module encodes the input graph and the pointer network finds the correct order of nodes. Finally a decoder module generates the surface form of tokens. The final approach can be used in other NLG pipelines to generate text.

    To the best of our knowledge, this is the first attempt to use transformers and pointer networks for the task of Surface Realization.

(2) We proposed an approach to take advantage of pretrained text-to-text LM for the task of Surface Realization.

    We represent the input graph in a text format and then feed it to the LM as the input and we use the surface form of the sentence as the target output to train the LM. The LM model learns to perform all intermediate steps (i.e. encoding, ordering, and inflection) at the same time.

    To the best of our knowledge, this is the first attempt to use text-to-text LMs for the task of surface realization.

(3) We proposed an approach for the task of Sentence Ordering that is able to find the most coherent order of a given set of sentences.

    In our approach, in contrast with previous approaches where sentences are encoded individually regardless of their position, we propose to encode sentences not only based on their content, but also based on their position. This new sentence representation which is conditioned on both the content of the sentence and the content of previously selected sentences is able to better capture the relation between sentences and hence improve the performance of the model. To the best of our knowledge, this is the first attempt to consider the position of sentences to create sentence embedding. The proposed model can be used for any NLG task that requires generating more than one sentence. Also, this approach can be used for tasks where the coherence of written text should be measured, such as essay evaluation.

(4) We proposed an approach to improve the performance of current surface realization approaches without the need to increase the amount of manually labeled training data.

    In this approach we take advantage of transfer learning by first training a model using synthetic data and then fine tuning it on the target dataset. We have created the synthetic dataset by parsing raw texts. Although the dataset created could be noisy, it sheer size helps the model to learn the task of surface realization. Then, by fine-tuning on the manually labelled data, we adapt the model to the distribution of the target dataset. Using different sizes of manually labeled data to fine-tune the model, we measured the effectiveness of our approach.

This thesis will show that the proper use of novel techniques such as pointer networks, conditional sentence representations, transfer learning, and text-to-text language modeling for specific NLG sub-tasks can improve the quality of generated text as well as provide new insight for modeling text coherence in computational linguistics.

Chapter 2

Literature Review

2.1 Introduction

As indicated in Section 1.3, our goal is to design models for surface realization and sentence ordering. In this chapter, we will study previous work done in this area as well as building blocks for our models.

2.2 Background

In this section, we will study related background for our proposed models.

2.2.1 Transformers

Transformers (Vaswani et al., 2017) introduced a new encoder-decoder architecture based self-attention for machine translation. In self-attention, for each element in the input sequence, we compute a representation using the attention mechanism where the query is the current element and keys and values are all the elements in the input sequence. This way, the obtained representation has the information of the current element as well as the information that relates this element to the other elements in the input sequence. The encoder consists of several encoder blocks where each block contains self-attention followed by a position-wise feed forward layer. The decoder transformer is similar to the encoder with two main differences in the decoder blocks. First, the decoder has a cross-attention alongside self-attention to get the information from the encoder. Second, in its self-attention, each element only attends to its own elements that appeared before itself in the sequence. This way, during inference, the decoder can generate the output step by step in an auto-regressive fashion.

After the success of transformers in machine translation, several works proposed the use of transformers for language modeling. For example, BERT (Devlin, Chang, Lee, & Toutanova, 2019) is a Language Model (LM) that uses encoder transformers; while GPT (Mager et al., 2020) uses the decoder part of the transformer. In addition, several works such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) use the encoder-decoder architecture.

Given the success of transformers in many NLP tasks (Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020), we have used this architecture in our proposed models.

2.2.2 Language Models for Data-to-Text

Data-to-text aims at generating text given an input provided in the form of graphs, tables, etc. As a family of data-to-text models, graph-to-text generation tries to generate natural texts given an input graph. Graph-to-text generation models employ graph encoders to obtain a suitable representation from the input graph. Several applications such as text summarization (Duan, Tang, Chen, & Zhou, 2017), question answering (Fan, Gardent, Braud, & Bordes, 2019), as well as surface realization (Du & Black, 2019) have used these types of generation models.

Transformer based language models such as BERT achieved the state-of-the-art (SOTA) in many NLP tasks (Devlin et al., 2019; Lewis et al., 2020). BERT is an encoder language model that encodes an input sentence(s) to a dense representation which can be used in downstream tasks. On the other hand, generative language models such as GPT (Mager et al., 2020) or BART (Lewis et al., 2020) can generate text from their inputs (e.g. Question Answering where the question is the input and the answer is the output).

Recently, several works have proposed to use language models in the form of text-to-text for tasks that they are inherently data-to-text and particularly graph-to-text tasks (Harkous, Groves, & Saffari, 2020; Kale & Rastogi, 2020; Mager et al., 2020). Instead of modeling the node and edges of the graphs, they map the graph structure as sequences of words and let the language model encode the graph information. Kale and Rastogi (2020) takes advantage of T5 (Raffel et al., 2020) for data-to-text problems, whereas Mager et al. (2020) and Harkous et al. (2020) utilize GPT2 (Radford et al., 2019).

Considering the success of text-to-text LM in different data-to-text generation task (e.g. Harkous et al. (2020); Kale and Rastogi (2020); Mager et al. (2020)), we investigated their effectiveness for the task of surface realization.

2.2.3 Pointer Networks

Pointer Networks (Vinyals, Fortunato, & Jaitly, 2015) are neural network models that are designed to sort variable sized sequences. Vinyals, Fortunato, and Jaitly (2015) showed that pointer networks have the ability to solve problems such as the Convex Hull or the Traveling Salesman Problem. Pointer networks builds upon general sequence-to-sequence models with attention mechanism (Bahdanau, Cho, & Bengio, 2015). In sequence-to-sequence models, an attention mechanism to compute the next input representation to the decoder. However, in a pointer network, attention is used as a pointer to select the next input to the decoder. Although originally, a pointer network referred to a sequence-to-sequence model, in later work including this thesis, it refers to a more general attention mechanism that chooses among a set of input. Hence, we can apply a pointer network on top of different architectures such as transformers or LSTMs.

Recently, pointer networks became the standard solution for sentence ordering tasks (Cui, Li, Chen, & Zhang, 2018; Logeswaran et al., 2018). This is why we investigate their performance for the task of surface realization since ordering is one of the steps in this task (see Section 1.2.2).

2.3 Sentence Ordering

Traditionally, sentence ordering approaches have relied on handcrafted linguistic features to model textual coherence. The Probabilistic Model (Lapata, 2003) used linguistic features to represent sentences in a vector space, then calculated the probability of transition between adjacent sentence vectors. The Content Model (Barzilay & Lee, 2004) modeled topic transition using Hidden Markov

Model (HMM), while the Entity Grid approach (Barzilay & Lapata, 2008) modeled entities transition between adjacent sentences to capture local coherence. These models constituted successful non-neural approaches to sentence ordering; however, they have been shown to be highly domain specific, hence difficult to port across domains.

To address the portability issue, several data-driven approaches have been proposed such as window networks (Li & Hovy, 2014) and pairwise ranking (Chen, Qiu, & Huang, 2016). These models use neural networks to represent sentences in a vector space, then using another neural network, they assign the probability of being coherent to each window or pair of consecutive sentences. Using this approach the models can assign a probability score to each ordering. Although these methods are able to discriminate coherent texts from non-coherent texts, they are not efficient enough to address the task of sentence ordering, as they require the evaluation of each possible sentence ordering.

More recently, sentence ordering models have been developed using encoder-decoder models taking advantage of pointer networks (Vinyals, Fortunato, & Jaitly, 2015) (see Section 2.2.3). In these models, a sentence encoder initially provides a dense representation of each sentence regardless of its context in the paragraph. Then, a paragraph encoder creates a context vector to represent the paragraph as a whole. Finally, the decoder, implemented as a pointer network (see Section 2.2.3), decides which sentence should be selected next given the paragraph representation as well as the previously selected sentences. A variety of architectures have been explored to use as encoders and decoders. For example, Gong, Chen, Qiu, and Huang (2016) and Logeswaran et al. (2018) used LSTMs (Hochreiter & Schmidhuber, 1997) for both the sentence and paragraph encoders as well as for the decoder.

RNN-based models process the input sequentially and are therefore sensitive to the order of input sequences. In other words, encoding the sequences $A\ B$ and $B\ A$ using RNN-based models would have different representations. In the task of sentence ordering, encoding paragraphs (sets of sentences), RNN-based models take the order of input sentences into account, hence, the same input with different ordering would lead to different results. Cui et al. (2018) proposed the use of a transformer without positional encoding to ignore the order of input sequence as the paragraph encoder. Later, Hierarchical Attention Networks (Wang & Wan, 2019) proposed to use an LSTM network for encoding words and transformer encoders for encoding sentences and paragraphs. They also utilized a transformer decoder for the pointer decoder.

Some studies tried to take advantage of linguistic features to augment pointer network based models. In particular, Topic-guided models (Oh, Seo, Shin, Jo, & Lee, 2019) augment sentence representations with their topic representation built using Latent Dirichlet Allocation (LDA). The pointer network chooses the next sentence based on both the content of the sentences and their topic transition.

Apart from pointer based models, Ranking Networks (Kumar, Brahma, Karnick, & Rai, 2020) tried to tackle the problem of sentence ordering as a ranking problem. They assigned a ranking score to each sentence and then sorted these sentences with their rank. They took advantage of the pre-trained BERT model and transformer encoders for encoding sentences and paragraphs. In the decoding step, a feed-forward network is used to predict a relative score of each sentence as an indicator for their position in the paragraph.

One of the common characteristics of the above-mentioned models lies in their paragraph encoder (either transformers or LSTMs) where an encoder tries to capture the relation of the sentences using a single vector representation for each sentence. This means that a single sentence representation has to capture both the content of the sentence and its relation with other sentences which limits the capacity of the model to capture local and global dependencies in the paragraph. Several works have been proposed to better represent these local and global dependencies. Graph-based models (Yin et al., 2019) use a graph encoder to encode a paragraph, where the graph is derived from the linguistic relations between the sentences. The nodes in this graph represent sentences and two sentences are connected if they have common tokens that are either subject or object in the parse tree. BERSON (Cui, Li, & Zhang, 2020) augmented the sentence encoder with a hierarchical relational sentence encoder. They capture the relation between each pair of sentences by passing them to the pre-trained BERT model. The BERT model encodes the dependencies between these two sentences into a single vector. Using both the relational representation and sentence representation, the pointer network can better capture the dependencies among sentences. RE-BART (Basu Roy Chowdhury, Brahman, & Chaturvedi, 2021) used pre-trained BART (an encoder-decoder language model) for sentence ordering. They mark sentences with special tokens (e.g. $< S1 >, < S2 >, < S3 >$) and feed all sentences to the encoder and train BART to generate the correct order of these special tokens (e.g. $< S3 >< S2 >< S1 >$). RE-BART captures all dependencies and interactions among tokens in all sentences and results in better performance compared to previous work.

2.4 Surface Realization

The Multilingual Surface Realization (MSR) workshops have organised shared tasks aimed at bringing together researchers interested in surface oriented Natural Language Generation problems and share resources (Belz et al., 2020; Mille, Belz, Bohnet, Graham, & Wanner, 2019; Mille et al., 2018). In this section we will study previous work on Surface Realization, mainly those who participated in the MSR shared tasks.

The MSR shared tasks were to reconstruct the sentences given their dependency parse tree in the form of surface realization. They used Universal Dependency datasets (de Marneffe et al., 2014) and then removed the inflected form of tokens and their correct order. The participants were required to find the inflected form of tokens and their order to reconstruct the sentences.

AX Semantics (Madsack et al., 2018) finds the order of words in the sentence using a classification approach and then using a rule-based method inflects the words in the sentence. In order to find the correct order for each sub-tree in the parse tree, they calculate the probability of two words appearing next to each other for all permutations of words in that sub-tree, and then, using a bottom-up approach they order all words in the sentence.

IIT-BHU (Singh, Sharma, Chawla, & Singh, 2018) first finds the inflected form of words using an LSTM based encoder-decoder model where the input is the lemmatized form of a word and its features and the output is the inflected form of the word. Then using an n-gram language model, they find the most probable order of inflected words.

The ADAPT center (Elder & Hokamp, 2018) created a map from the Wiki dataset (Merity et al., 2017) to find the inflected from of words. Then using depth first search, they created the linearized version of the input parse tree to train a sequence-to-sequence LSTM model with copy attention.

CMU (Du & Black, 2019) used a Graph Attention Network to encode the input graph and using a pointer network with LSTM as the decoder, finds the correct order of words. In the end, using a character-level RNN language model, they find the inflected form of words.

BME-UW (Kovács, Ács, Ács, Kornai, & Recski, 2019) first found the correct order of words for each sub-tree in the input using Interpreted Regular Tree Grammar (IRTG) and then using a sequence-to-sequence model with a biLSTM encoder and an LSTM decoder with attention, they found the inflected form of words.

2.5 Conclusion

In this chapter, we described the background knowledge required for our work as well as previous work done for Sentence Ordering and Surface Realization. Sentence Ordering models currently achieve promising results but there is still room for improvement as they do not address the local dependencies among sentences. In chapter 5, we introduce cognitional representation to capture these local dependencies. On the other hand, the performance of Surface Realization (SR) models are below human level (Mille et al., 2018). This is mainly due to limited available labeled training data. In the next chapter, we investigate our first research question to see if a pointer network can address task surface realization task. We further investigate the ability of LM to address surface realization in Chapter 4 and in Chapter 6, we improve the performance of these models using transfer learning approaches.

Chapter 3

Surface Realization as an Ordering Problem

In this chapter we try to answer the first research question in Section 1.3: "Can we address the problem of surface realization as an ordering problem?".

Please note that the content of this chapter has been published at the 2nd Workshop on Multilingual Surface Realisation, MSR-2019 (Farahnak et al., 2019).

Abstract

This chapter presents the model we developed for the shallow track of the 2019 Natural Language Generation (NLG) Surface Realization Shared Task. The model reconstructs sentences whose word order and word inflections were removed. We divided the problem into two sub-problems: reordering and inflecting. For the purpose of reordering, we used a pointer network integrated with a transformer model as its encoder-decoder modules. In order to generate the inflected forms of tokens, a Feed Forward Neural Network was employed.

3.1 Introduction

The goal of NLG is to produce natural texts given structured data. Typically, NLG is sub-divided into two tasks: Content Planning and Surface Realization (Hovy, van Noord, Neumann, & Bateman, 1996; Reiter & Dale, 2000). While Content Planning focuses on selecting the most appropriate content to convey, Surface Realization produces the linear form of the text from this selected data following a given grammar.

Although the field of Natural Language Processing (NLP) has witnessed significant progress in the last few years, NLG, and surface realization in particular, still performs significantly below human performance (Mille et al., 2018).

Recently, several shared tasks have been proposed to improve the state of the art in specific NLG tasks (eg. Dušek, Novikova, and Rieser (2020); May and Priyadarshi (2017)). In particular, the Surface Realization Shared Task 2019 (SR'19) (Mille et al., 2019) aims to provide common-ground datasets for developing and evaluating NLG systems. Similarly to SR'18 (Mille et al., 2018), SR'19 proposed two tracks: a shallow track and a deep track. In the shallow track, unordered and lemmatized tokens with universal dependency (UD) structures (de Marneffe et al., 2014) were provided to participants and systems were required to reorder and inflect the tokens to produce final sentences. The deep track is similar to the shallow track but functional words and surface-oriented morphological information were removed as well. In addition to determining token order and inflections, systems participating in the deep track also had to determine the omitted words.

We decided to only participate in the shallow track. We used a model based on the transformer encoder-decoder architecture (Vaswani et al., 2017) combined with a pointer network (Vinyals, Fortunato, & Jaitly, 2015) to reconstruct the word order from the input provided and a Feed Forward Neural Network to produce inflections. Based on the human evaluation, our model has an average score of 48.1% and 60.9% on all the English datasets for Readability/Quality and Meaning Similarity respectively.

3.2 Related Work

Pointer networks are types of encoder-decoder models where the output corresponds to a position in the input sequences (Vinyals, Fortunato, & Jaitly, 2015). One of the main advantages of pointer networks compared to standard sequence-to-sequence models is that the number of output classes depends on the length of the input. This feature can be useful to address problems involving sorting variable sized sequences such as required at SR'19.

In Vinyals, Fortunato, and Jaitly (2015), Recurrent Neural Networks (RNNs) are used as encoder and decoder. RNNs compute the context representation based on the order of the input sequences. In cases where there is no information regarding the correct order of the input sequences, using an RNN-based encoder cannot provide a proper context representation for the decoder.

Transformer models constitute an alternative to RNNs as they entirely rely on the self-attention mechanism (Vaswani et al., 2017). Transformer models have achieved state of the art performance in many NLP tasks such as machine translation (Vaswani et al., 2017) and language modeling (Devlin et al., 2019; Radford et al., 2019).

The encoder and decoder modules of transformer models consist of multiple layers of stacked self-attention and point-wise fully connected layers. The encoder of the transformer consists in several encoder layers, each of which is composed of two sub-layers. The first sub-layer has a multi-head attention which consists of several heads of self-attention computing on the same input, and the second sub-layer is a feed-forward network. The output of each sub-layer is added with a residual connection from their input followed by a normalization layer. The decoder module consists of several layers similar to the encoder, where the decoder layers have an extra sub-layer of encoder-decoder attention.

Transformer models have no information regarding the order of the input sequence. Hence “Mary killed John” and “John killed Mary” have the same internal representations. To alleviate this issue, Vaswani et al. (2017) considered using positional encoding summed to the embedding of each word. Because the transformer without positional encoding does not rely on the order of the input sequence, this architecture constitutes a promising option for the SR'19 where the correct order of the input sequence were removed (see Section 3.3).

#	Feature	Feature description	Embedding size
1	`Token`	Lemma or stem of word form	300
2	`UPOS`	Universal part-of-speech tag	10
3	`XPOS`	Language-specific part-of-speech tag	10
4	`Deprel`	Universal dependency relation to the `Head`	10
5	`Head`	Head of the current word	20
6	`Index`	Word index	20
7	`Lin`	Relative linear order with respect to the governor	20
8	`Lin sign`	The sign of the `Lin` feature	3
All		Concatenation of all features	393

Table 3.1: The 8 features used in our model with their corresponding embedding sizes

3.3 Dataset

For the shallow track, training and development sets were provided for 11 different languages. These were taken from the Universal Dependency (UD) datasets (de Marneffe et al., 2014). The correct token order within the sentences was removed by shuffling the tokens. In total, 7 features were provided by the organizers. Out of these features, `FEATS` contained more than 40 morphological sub-features from the universal feature inventory and the relative linear order with respect to the governor (`Lin`). Table 3.1 lists the 8 features used by our model: 6 features of the UD structure, in addition to 2 features for `Lin` (the `Lin` feature divided into its absolute value and its sign).

In particular, we worked only on the English datasets, which consists of four training and development pairs. We concatenated all four training sets (`en_ewt-ud`, `en_gum-ud`, `en_lines-ud` and `en_partut-ud`) into a single one containing 19,976 sentences, with the longest sentence containing 209 words.

Because the development sets provided by the SR'19 organizers are not labeled, we divided the training data in two parts; training (18, 000 sentences) and validation (1, 967 sentences). We removed all sentences longer than 100 tokens for efficiency reasons.[1]

[1]This removed 9 sentences from the training sets.

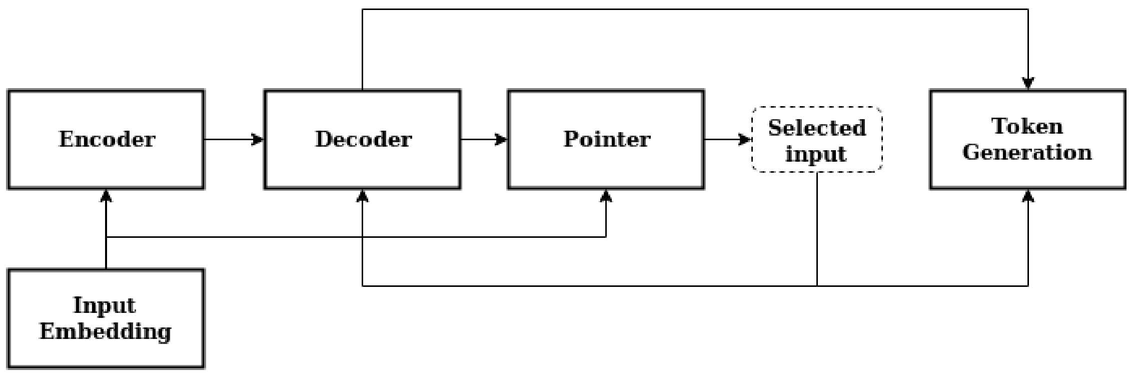


Figure 3.1: Model architecture used for the shallow track at SR'19

3.4 Model

Inspired by sentence ordering models Cui et al. (2018); Logeswaran et al. (2018) that used pointer networks to reconstruct unordered sentences, we developed a similar model using a pointer network integrated with a transformer as its encoder and decoder. As shown in Figure 3.1, our model is composed of five modules: input embedding, encoder, decoder, pointer, and token generation. In the following, we describe each module in more detail.

Input embedding In order to train the model, we embedded each feature separately into vectors and then concatenated them. Table 3.1 indicates the embedding size of each feature. For the token embeddings, we employed the GloVe pretrained embeddings of size 300 (Pennington, Socher, & Manning, 2014), while the remaining feature embeddings are trained from scratch. At the end, the concatenated vector (of size 393) is linearly mapped into the desired embedding size (512 in our case, see Section 3.5).

3.4.1 Encoder

The embedded input is fed into the encoder to compute its representation. The encoder module uses the transformer architecture described in Vaswani et al. (2017). Since the input data does not provide any ordering information, we directly feed the embedded input into the encoder without summing it with the positional encoding of the tokens.

3.4.2 Decoder

The decoder in the transformer model receives the previously generated tokens alongside the encoded representation from the encoder as its input to generate the next token. However, since our task is to produce an ordering rather than generate tokens, we decided to feed the same embeddings used for the encoder (see Section 3.4) in its correct order. Since the correct order for the previously generated tokens is determined in the decoding phase, we add the positional encoding with the embedded input.

3.4.3 Pointer

To find the next token, we deploy the attention mechanism described in Vaswani et al. (2017) and in Equation 1.

$$\text{Attention}(Q, V, K) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

In each decoding step, the keys (K) and values (V) come from the embedded input and the query (Q) is defined by the output of the decoder (in this setup the keys and query have a dimension of d_k). The pointer selects the most probable embedded input as the next input to the decoder. During test time, we mask out the previously selected embedded inputs so that they are not selected again.

3.4.4 Token Generation

The Pointer Network orders the given lemmatized tokens based on the input features. However, the desired output should be the inflected form of the input tokens. To this end, the token generation module (see Figure 3.1) is designed to generate the inflected form of the tokens, where its input is the concatenation of the selected embedded input token with the decoder's output. The output of this module is the probability over all the words in the vocabulary. This module consists of two feed-forward layers with a ReLU activation function. The last layer is initialized with pretrained GloVe embeddings in order to provide a better generalization on unseen tokens.

			Tokenized			Detokenized		
#		Dataset	BLEU	NIST	DIST	BLEU	NIST	DIST
1	In-domain	en_ewt-ud-test	**22.08**	9.77	45.99	14.62	7.21	44.7
2		en_gum-ud-test	15.32	8.64	38.13	10.55	6.53	36.97
3		en_lines-ud-test	15.30	8.23	40.40	9.81	6.10	39.08
4		en_partut-ud-test	10.07	7.14	36.21	7.45	5.57	35.32
5	Out-of-domain	en_pud-ud-test	12.36	8.83	36.26	9.11	6.66	35.23
6	Predicted	en_ewt-Pred-HIT-edit	21.21	9.69	43.59	14.14	7.23	42.41
7		en_pud-Pred-LATTICE	12.89	8.82	36.67	9.29	6.69	35.53

Table 3.2: Results of our submission in the shallow track task of SR'19

3.5 Experiments and Results

3.5.1 Model Configuration

The model submitted to SR'19, under the team name CLaC, has the following configuration optimized on the validation set.

The encoder and decoder of the model have 4 layers each with 8 heads (number of attention in each layer). All the embedding sizes of the encoder and decoder layers as well as input embedding are set to 512.

To preserve the GloVe embeddings, we froze the weights of both token embeddings and the last layer of the token generation module.

We suspected that the `Head` and `Index` features (see Table 3.1) constituted valuable information regarding the dependency tree structure of the sentences. Therefore, to ensure the model does not memorize the actual value of these features, but rather their relationship, in each training iteration, we randomly changed the values of these features while keeping the tree structure relationship intact.

The model was trained using two cross-entropy loss functions: order loss for the pointer and token loss for the generation module. We observed training with the order loss and then fine tuning on both losses increased the performance of the model.

The final model trained for 60 epochs, where training on the order loss is done for 30 epochs, and fine tuning the order and token losses were done on the remaining 30 epochs. The initial learning rate was set to 1×10^{-4}. We also took advantage of learning rate decay with the factor of 0.5 when there is no improvement on the validation loss. A dropout rate of 0.3 was used on the encoder, decoder, and input embedding module.

The model was implemented using the PyTorch 1.1 framework. For the transformer encoder and decoder, we modified the fairseq transformer implementation of Ott et al. (2019).

In the test phase, we used tokens generated by the model as the final output. When encountering unknown tokens, the model uses the input token where the pointer points at.

3.5.2 Results

At SR'19, three types of test sets were given: In-domain, Out-of-domain, and Predicted. The In-domain datasets share the same domain as the training data, while Out-of-domain dataset does not. The Predicted datasets are those where the annotation were built using parser outputs from the Universal Dependency Parsing shared task 2018 (Zeman et al., 2018) instead of the gold syntactic annotations. Evaluation was performed in both a tokenized and detokenized fashion.

Table 3.2 shows the results of our model on the test data. As shown in Table 3.2, our model achieved its highest performance on the `en_ewt-ud-test` dataset with a BLEU score of 22.08 for tokenized among the In-domain datasets. Whereas the lowest score is for `en_partut-ud-test` with a BLEU of 10.07. Clearly, the performance of the model on these four datasets is directly related to the relative size of corresponding training set in the concatenated training set used to train our model. For example, the `en_ewt-ud-train` dataset accounts for the greatest proportion of our training set (63%) and achieves the highest BLEU; whereas `en_partut-ud-train` accounts for only 9% of the training samples yielding the lowest BLEU of 10.07 with `en_partut-ud-test`.

Based on human evaluation, our submitted system achieved average Readability/Quality score of 48.1% and a Meaning Similarity score of 60.9% with the rank of 12 and 14 respectively among the 16 participating systems. The results are shown in the Figure 3.2.

3.6 Conclusions

In this chapter, we have presented the model we developed for SR'19. The proposed system is composed of a pointer network where its encoder and decoder modules borrowed from transformer, aim to reconstruct the tokens' order and inflection. The model achieved its best performance on the English datasets with the average scores of 48.1 and 60.9 for the Readability/Quality and Meaning

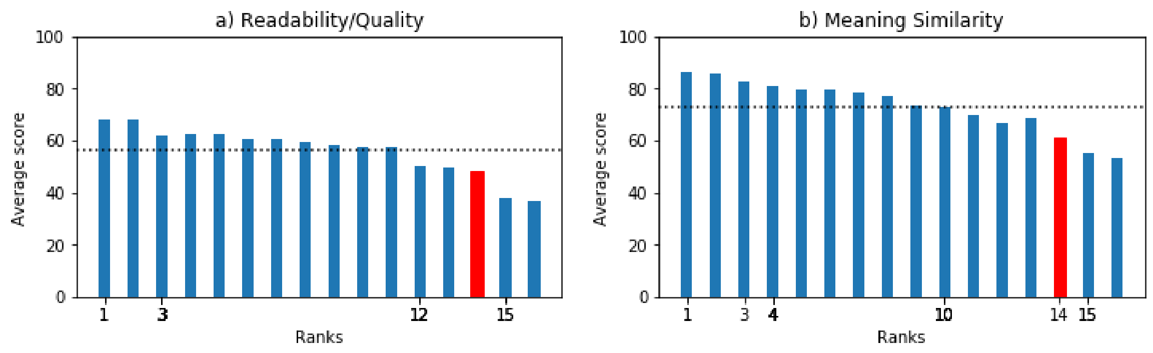


Figure 3.2: Human evaluation results compared to all participants of the shallow track at SR'19

Similarity respectively. Although this performance is lower than expected, the lack of training data is observable. It was noticeable that training the model in an end-to-end fashion without feature engineering could not lead the model to learn meaningful representation of the input features.

Chapter 4

Surface Realization Using Pre-trained Language Models

In the previous chapter, we presented a Pointer Network for surface realization to address the Surface Realization as an ordering problem. We showed that the Pointer network can find the correct order of tokens and the inflection module can guess the inflected form of them. However, the overall performance of the model was below the state-of-the-art and we need to revisit our approach. In this chapter, we tackle the Surface Realization as a token generation problem and use a pre-trained text-to-text Language Model. In this chapter we try to answer the second research question in Section 1.3: "Can we use a simple language model to solve the task of surface realization?".

Please note that the content of this chapter has been published at the Third Workshop on Multilingual Surface Realisation, MSR-2020 (Farahnak et al., 2020).

Abstract

In the context of Natural Language Generation, surface realization is the task of generating the linear form of a text following a given grammar. Surface realization models usually consist of a cascade of complex sub-modules, either rule-based or neural network-based, each responsible for a specific sub-task. In this work, we show that a single encoder-decoder language model can be used in an end-to-end fashion for all sub-tasks of surface realization. The model is designed based on the

BART language model that receives a linear representation of unordered and non-inflected tokens in a sentence along with their corresponding Universal Dependency (UD) information and produces the linear sequence of inflected tokens along with the missing words. The model was evaluated on the shallow and deep tracks of the 2020 Surface Realization Shared Task (SR'20) using both human and automatic evaluation. The results indicate that despite its simplicity, our model achieves competitive results among all participants in the shared task.

4.1 Introduction

Natural Language Generation (NLG) models aim to generate fluent human-like texts given structured data. This involves both *content planning* (selecting the content to communicate) and *surface realization* (selecting, ordering, and inflecting the actual words) (Hovy et al., 1996; Reiter & Dale, 2000). This chapter focuses on the second sub-task: Surface Realization (SR).

Unlike many tasks in Natural Language Processing (NLP), the performance of SR models are still below human performance. In order to fill this gap and encourage more research in this field, several shared tasks in NLG and SR have been proposed. In particular, since 2018, the Surface Realization Shared Tasks (Belz et al., 2020; Mille et al., 2019, 2018) were introduced to provide common-ground datasets for developing and evaluating NLG systems. This year, the task (Belz et al., 2020) proposed two tracks in several languages including English: 1) Track1: *shallow track* and 2) Track2: *deep track*. In the shallow track, unordered and lemmatized tokens with UD structures (de Marneffe et al., 2014) were provided to participants and systems were required to reorder and inflect the tokens to produce final sentences. The deep track was similar to the shallow track but functional words and surface-oriented morphological information were not provided and had to be inferred by the systems. Therefore, in addition to determining the order and the inflection of tokens, systems participating in the deep track had to guess the omitted words.

Considering that the input data is in the form of UD structure, data-to-text models, and graph-to-text models in particular, seem to be the right choice for the task of surface realization. However, in this study, we take a different path to tackle the problem. Our proposed model is designed based on text-to-text approaches using a pretrained encoder-decoder language model. More specifically, a

BART (Lewis et al., 2020) language model is used for the task of surface realization for both the shallow and the deep tracks. The proposed approach is an end-to-end model trained on a linearized representation of the graph of the sentences with their corresponding UD information. The results on the English datasets demonstrate the potential of these models for surface realization tasks and in general data-to-text problems.

In the following sections, we first review recent and related approaches in surface realization and text-to-text models, then in Section 4.3, our proposed model is explained. Section 4.4 describes the results of our model on the surface realization datasets. Finally, Section 4.5 discusses conclusion and future work.

4.2 Related Work

Surface realization typically involves three tasks: syntactic realization, morphological realization, and orthographic realization (Reiter & Dale, 2000). Syntactic realization tries to identify the proper ordering of the input data, whereas morphological and orthographic realization are responsible for word inflections, punctuation, and formatting.

Several surface realization models presented at the previous Surface Realization Shared Task (Mille et al., 2019) used a cascade of pointer-based models for syntactic realization followed by another neural network module for morphological and orthographic realization (Du & Black, 2019; Farahnak et al., 2019; Mazzei & Basile, 2019). For example, Du and Black (2019) utilized a graph attention network (GAT) (Veličković et al., 2018) for encoding the input sentences and a pointer decoder (Vinyals, Fortunato, & Jaitly, 2015) to select the next element from their graph. Whereas Yu, Falenska, Haid, Vu, and Kuhn (2019) used a bidirectional Tree-LSTM (Zhou, Liu, & Pan, 2016) as the encoder and an LSTM (Hochreiter & Schmidhuber, 1997) as the decoder and multiple LSTM modules for morphological and orthographic realization tasks. These two approaches achieved the highest performance among all participating systems at SR'19.

On the other hand, when the text to generate is conditioned on the content provided in the form of graphs, tables, etc then data-to-text generation models are utilized. As a family of data-to-text models, graph-to-text generation tries to generate natural text given its input graph. Graph-to-text generation models employ graph encoders to obtain a suitable representation from the input graph. Several applications such as text summarization (Duan et al., 2017), question answering (Fan et al., 2019), as well as surface realization (Du & Black, 2019) have used these types of generation models.

Recently, several works have proposed to use language models in the form of text-to-text for what is inherently data-to-text and particularly graph-to-text tasks (Harkous et al., 2020; Kale & Rastogi, 2020; Mager et al., 2020). Instead of modeling the node and edges of the graphs, they mapped the graph structure as sequences of words and let the language model encode the graph information. Kale and Rastogi (2020) takes advantage of T5 (Raffel et al., 2020) for data-to-text problems, whereas Mager et al. (2020) and Harkous et al. (2020) utilize GPT2 (Radford et al., 2019).

Following this recent trend of text-to-text models for graph-based problems, we developed an end-to-end approach based on a language model for the problem of data-to-text generation. The approach maps the given UD structures to surface forms to tackle all the tasks required for the surface realization.

4.3 Model

Following the success of pretrained language models for data-to-text generation (Harkous et al., 2020; Kale & Rastogi, 2020), we tackled the surface realization problem using a similar approach. BART (Lewis et al., 2020) is an encoder-decoder language model based on transformers (Vaswani et al., 2017). Specifically, BART is a denoising autoencoder model trained on several denoising tasks making it applicable for a variety of downstream NLP tasks. The language model is trained to first encode the input, and then generate the text based on its input representation. The encoder-decoder architecture of BART makes it suitable for our task where the encoder module encodes the graph representation into an embedding space then the decoder generates the inflected form of the input one token at each decoding step. Hence, the model performs syntactic, morphological, and orthographic realization all at the same time.

In order to take advantage of a pretrained language model, BART in our case, we first need to map the graph structure of the input into a linear representation (i.e. plain text). For this task, given all the UD information provided by the SR'20 organizers, we considered using only `LEMMA`,

```
      ...
(a)    26     kill     killed  VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin|original_id=7  20      parataxis      _      _
      ...

(b)                            <s> LEMMA < HEAD > UPOS | FEATS | DEPREL <\s>

(c)                        ... <s> kill < Al > VERB | Ind Past Fin | parataxis <\s> ...
```

Figure 4.1: Sample token with its linearized representation. The sample token with index 26 and its UD information in (a) will be encoded using (b) to generate its linearized representation in (c).

`UPOS`, `FEATS`, `HEAD`, and `DEPREL` for each node. Figure 4.1 shows the mapping structure with an example of a linearized node. As Figure 4.1 shows, we concatenate each token with its corresponding features where special tokens `<`, `>`, `|` identify the boundary of each feature and special tokens such as the beginning of sentence `<s>` and end of sentence `<\s>` determine the boundary of each node (i.e. token). In these mapped representations, instead of the index for the `HEAD` feature, the actual token of the `HEAD` is used.

As opposed to previous work where tokens are selected among the input (Du & Black, 2019; Mazzei & Basile, 2019), the BART decoder generates tokens at each decoding step. As a result, we have no control on the number of tokens that should be generated. This can lead the model to generate extra tokens or ignore some of the input tokens. To alleviate this issue, we considered the tokenized form of the target sentences as the targets of the model. This allows the model to learn to map each node representation in its input into a token in the output and achieve a 1:1 mapping between the number of nodes in the input and the number of output tokens. An alternative would be to train on the detokenized output form but for the sake of automatic evaluation of the shared task, we did not follow this option.

4.4 Experiments and Results

4.4.1 Datasets

The proposed text-to-text surface realizer model was evaluated on the English datasets of the SR'20 (Belz et al., 2020). The datasets were created using the UD datasets (de Marneffe et al., 2014) where the tokens within each sentence were randomly shuffled and the inflections were removed. The training and development sets were accompanied by 7 features. Out of these, the `FEATS` feature contained the relative linear order with respect to the governor (`Lin`) in addition to more than 40 morphological sub-features from the universal feature inventory. Among the four English datasets provided for training, we only trained on `en_ewt-ud-train` while the performance of the model was measured on all the eight test sets provided by the organizers (see Table 4.1).

4.4.2 Model Configuration

We used the pretrained BART-large model provided in the Huggingface library (Wolf et al., 2019). This model has 12 layers in each encoder and decoder modules. For each task, we trained our models with the same cross-entropy loss function as suggested in the original paper of BART and AdamW algorithm (Loshchilov & Hutter, 2019) with batch size of 2 and the learning rate of $1e-5$ for 20 epochs where these hyper-parameters are chosen based on the development set. For efficiency in training time, sentences longer than 35 tokens were removed from the training set. During inference, we used beam search with a size of 5.

4.4.3 Results

			T1				T2			
#		**Dataset**	**BLEU**	**NIST**	**DIST**	**BERTScore**	**BLEU**	**NIST**	**DIST**	**BERTScore**
1	In-domain	`en_ewt-ud`	70.71	12.70	77.94	0.9565	58.44	11.61	73.66	0.9635
2		`en_ewt-Pred`	67.12	12.40	74.47	0.9562	56.01	11.19	69.00	0.9593
3	Out-of-domain	`en_pud-ud`	74.47	12.62	76.46	0.9605	**58.45**	11.43	68.83	**0.9613**
4		`en_pud-Pred`	73.41	12.52	77.68	0.9596	**56.69**	11.18	67.79	**0.9584**
5		`en_gum-ud`	66.98	11.62	69.87	0.9555	53.92	10.51	67.02	0.9572
6		`en_lines-ud`	62.70	11.30	68.62	0.9565	47.96	9.93	64.33	0.9487
7		`en_partut-ud`	67.05	9.83	71.59	0.9565	**50.54**	8.57	62.39	0.9505
8		`en_Wikipedia`	74.66	12.90	76.93	0.9614	**57.49**	11.57	67.26	0.9473

Table 4.1: Results of our submission in the shallow (T1) and deep (T2) tracks of SR'20. Highlighted values indicate the highest score among all participants achieved by our model.

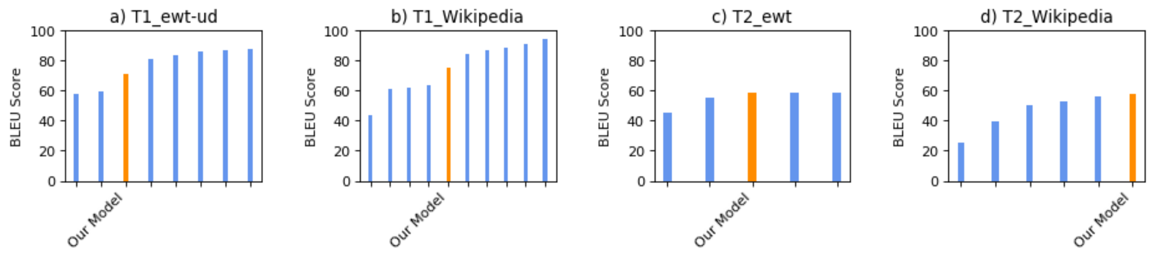


Figure 4.2: Comparison of all systems submitted to SR'20 on the `en_ewt-ud` and `en_Wikipedia` test sets for the shallow (T1) and the deep (T2) tracks evaluated based on their BLEU score.

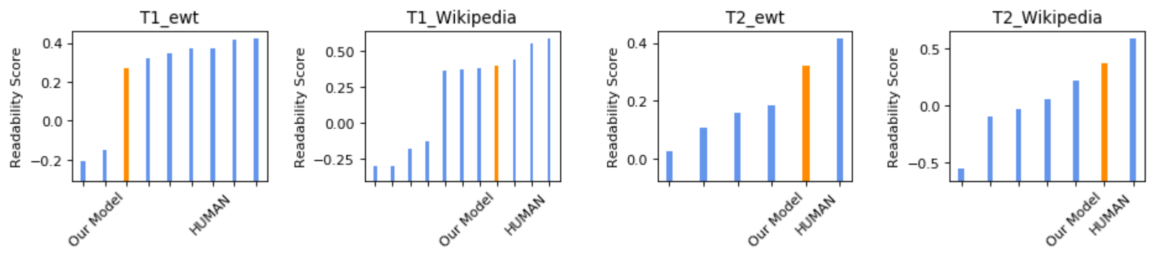
Our surface realization model was evaluated on all seven test sets from SR'19 and one new test set (`en_Wikipedia`) for SR'20. Results with all test sets are presented in Table 4.1 where *in-domain* refers to set-ups where test sets and training sets are from the same domain, while *out-of-domain* refers to set-ups where test sets are not in the same domain as the training sets. Two types of test sets were provided: *ud* and *pred*. The *ud* test sets contain the original UD datasets; whereas the *pred* test sets contain automatic predicted parse trees. Since we only trained our model on `en_ewt-ud-train`, all the other test sets should be considered out-of-domain for our model.

Our models were evaluated both with automatic metrics and human evaluation. For the automatic metrics, as shown in Table 4.1, each test set was evaluated using four different metrics: BLEU, NIST, DIST, and BERTScore.

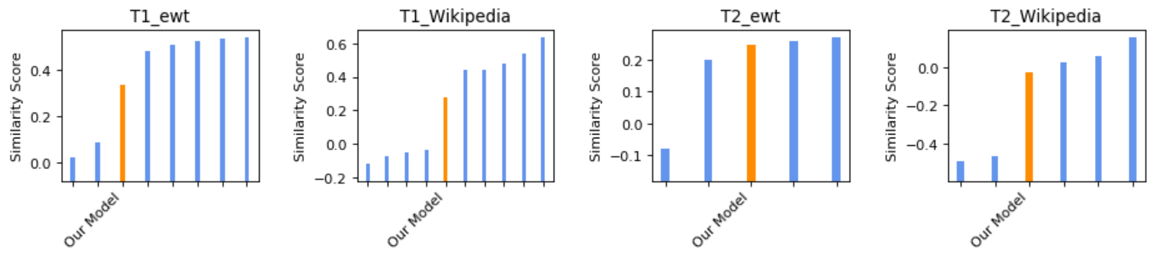
Figure 4.2 compares the BLEU score of our model with all other participants with the test sets `en_ewt-ud` and `en_Wikipedia` for both shallow (T1) and deep (T2) tracks. As Figure 4.2 shows, our model's performance falls close to the median of the other models in the shared task for the shallow track. However, in the deep track, our model performs best (Figure 4.2d) or close to the best (Figure 4.2c). These results seem to indicate that BART can better predict functional words and surface-oriented morphological information (required only in deep track) and this is mainly due to the denoising approaches used for training BART.

For human evaluation, SR'20 used Direct Assessment (Graham, Baldwin, Moffat, & Zobel, 2017) where human assessors rated the output of systems based on meaning similarity (relative to a human-authored reference sentence) and readability (without reference sentences). Figure 4.3 shows the average score for the meaning similarity and readability on both shallow and deep tracks on two test sets: `en_ewt-ud` nad `en_Wikipedia`. In terms of readability, in the deep track, our model achieved the highest score after Human (see Figure 4.3a). However, in the shallow track, our performance is at the median of all participants. On the other hand, the meaning similarity of our model is lower than the best performing model (see Figure 4.3b). These comparisons indicate that although our model is able to generate well-written texts, their meaning are not close enough to the target sentences.

In order to have a better understanding of the model's behaviour, we further analyzed the generated outputs of our model. Our decoder generates tokens from its encoder hidden space, which



(a) Readability



(b) Meaning Similarity

Figure 4.3: Human evaluation comparison of systems submitted in SR'20 on the `en_ewt-ud` and `en_Wikipedia` test sets for shallow (T1) and deep (T2) tracks evaluated based on their (a) readability and (b) meaning similarity.

is in contrast with ordering models where the output is selected from their input. This characteristic can have two impacts: A) the model does not restrict the output to the words in the input and it has no control over the number tokens it should generate. As a result, the model can generate extra tokens or ignore tokens from the input. Example A in Table 4.2 highlights this issue where the model did not generate the token *Ali*. B) the model generates long tokens in more than a single decoding step which makes it too complex for the model to have control over the details of such tokens. This behaviour is shown in Table 4.2 Example B. In this example, even though the model was able to generate a very complex URL similar to its input, there are some missing parts such as *27* or even extra parts such as *.html*. It should be noted that in Example B, the entire URL is considered as a single token.

4.5 Conclusion

In this chapter, we used a language model (text-to-text) to tackle the graph-to-text surface realization problem. We showed that using a pretrained encoder-decoder language model such as BART, can allow the model to reconstruct the tokens' orders and inflections in an end-to-end fashion

A	Target	Shazad Khuram Ali , 27 , High Wycombe
	T1 output	Shazad Khuram , 27 High Wycombe ,
	T2 output	Shazad Khuram 27 High Wycombe Ali
B	Target	I hope you do n't mind , but I 've taken liberty to turn them into a web photo album at http://24.27.98.30/pictures/08-05_Garrett_Gayle_Bday .
	T1 output	I hope you do n't mind , but I 've taken liberty to turn them into a web photo album at http://24.98.30/pictures/08-05_garrett_gayle_bday .
	T2 output	I hope you do n't mind but I have taken liberty to turn them into a web photo album (http://24.98.30/pictures/08-05_garrett_gayle_bday.html) .

Table 4.2: Examples of unsuccessful outputs generated by our model on the shallow (T1) and deep (T2) tracks.

without any modification in the network architecture. This shows that the pretrained language model is able to encode the graph information and map this representation into a human readable text. The results of the deep track showed that our proposed model achieved better performance compared with the systems submitted to SR'20 (Belz et al., 2020) where the model has to guess the functional words and morphological information. This high performance directly comes from the pretrained language model characteristics of BART.

Chapter 5

Sentence Ordering via Pointer Networks

The two previous chapters addressed the Surface Realization problem. In this chapter we try to answer the third research question formulated in Section 1.3: “Can we design a sentence representation that captures the relation among sentences?”. The idea is to use a Pointer Network for Sentence Ordering and enhance it with a conditional sentence representation to better capture the relation among sentences.

Please note that the content of this chapter has been published at the 15th International Conference on Semantic Computing, ICSC-2021 (Farahnak & Kosseim, 2021).

Abstract

Sentence ordering aims to arrange sentences in a coherent manner and hence has important applications in Natural Language Generation. Recently, several approaches have used Pointer Networks for this task. Such networks arrange a list of sentences based on fixed sentence representations, where these representations are independent of the sentence’s position in the text and its relation to the previously selected sentences. In this work, we propose a conditional sentence representation, which incorporates the information of the previously selected sentences into the candidate sentence representations. By using such information, the Pointer Network is able to better capture dependencies among sentences. Experiments indicate that our proposed model achieves state-of-the-art performance on most sentence ordering benchmarks and achieves a significant improvement over

state-of-the-art performance on short stories datasets.

5.1 Introduction

Achieving textual coherence is a key goal of natural language generation. In order to be coherent, a text needs to have a clear discourse structure that makes it easy for the reader to understand the relations between its elements and their communicative goals. Several work has addressed the modeling of text coherence in natural language processing (e.g. (Barzilay & Lapata, 2008; Hobbs, 1990; Mann & Thompson, 1988)). A particular sub-task of the generation of coherent text is sentence ordering, whose aim is to arrange a given set of sentences into its most coherent order. Sentence ordering has attracted much attention in recent years (e.g. (Chen et al., 2016; Cui et al., 2018; Logeswaran et al., 2018)) and is useful for several downstream applications such as multi-document text summarization (e.g. (Barzilay et al., 2001; Nallapati et al., 2017)) or concept-to-text generation(e.g. (Konstas & Lapata, 2012)).

Recently, several methods have proposed the use of pointer networks (Vinyals, Fortunato, & Jaitly, 2015) for sentence ordering (e.g. (Cui et al., 2018; Logeswaran et al., 2018)). One key advantage of these models is that they are able to predict the order of the set of sentences directly, as opposed to computing and evaluating all possible sentence orderings, hence significantly reducing their run-time requirements while achieving competitive performance.

In a coherent discourse, sentences are connected to one another not only based on their main content but also on more subtle aspects that are highlighted when the sentence is placed in a specific context. Although sentence representations that are independent of their position and their surrounding sentences can capture the main content of each sentence, secondary aspects which can further link consecutive sentences logically are not well captured. In the task of sentence ordering, we do not have access to the correct positional information of sentences beforehand, hence in conventional pointer-based models, the sentence representations are the same regardless of the time step and their position. We argue that, at each decoding step, the decoder should consider different aspects of the sentences depending on the previously selected sentences in order to better capture the discourse relations between consecutive sentences. We believe that it is worthy of exploring a sentence representation that not only is a function of the content of the sentence but also is a function of the candidate position of the sentence and the surrounding sentences in their correct order for sentence ordering.

In order to better model discourse coherence, we have developed a conditional sentence representation that captures local dependencies among consecutive sentences. These representations are obtained based on the content of the sentence conditioned on the information of previously selected sentences. At each decoding step, the pointer is looking for different information based on previously selected sentences. Therefore, at each decoding step, the model creates a new conditional representation for the candidate sentences. Considering these representations alongside conventional static representations allows the model to better capture logical and structural dependencies for sentence ordering.

In this chapter, we propose to use a conditional sentence representation in pointer networks for the task of sentence ordering. Our model consists of an encoder and a decoder. The encoder is a hierarchical attention mechanism stacked on top of a pre-trained BERT model (Devlin et al., 2019). The pointer decoder uses the proposed dynamic sentence representations to predict the correct order of sentences in a paragraph. We calculate these conditional representations considering both the content of the sentence and the previously selected sentences in the text. Experiments show that our model achieves state-of-the-art performance on several sentence ordering benchmarks. It achieves significantly better performance on stories datasets where the local relation of consecutive sentences plays a pivotal role in the task.

In the following sections, we first present previous work in the area, then in Section 5.3 we introduce our model in detail. Section 5.4 describes the results of our models on different datasets along with analysis of the results. Finally, Section 5.5 discusses conclusion and future work.

5.2 Related Work

Traditionally, sentence ordering approaches have relied on handcrafted linguistic features to model textual coherence. The Probabilistic Model of (Lapata, 2003) used linguistic features to represent sentences in a vector space, then calculated the probability of transition between adjacent sentence vectors. The Content Model of (Barzilay & Lee, 2004) modeled topic transition using Hidden Markov Model (HMM); while the Entity Grid approach of (Barzilay & Lapata, 2008) modeled entities transition between adjacent sentences to capture local coherence. These models constituted successful non-neural approaches; however, they have been shown to be highly domain specific, hence difficult to port across domains.

To address the portability issue, several data-driven approaches have been proposed such as window network (Li & Hovy, 2014) and pairwise ranking (Chen et al., 2016). These models used neural networks to represent sentences in a vector space, then assigned the probability of being coherent using another neural network. Although these methods are able to discriminate coherent texts from non-coherent texts, they are not efficient enough to address the task of sentence ordering, as they require the evaluation of each possible sentence ordering.

More recently, sentence ordering models have been developed using encoder-decoder models taking advantage of pointer networks (Vinyals, Fortunato, & Jaitly, 2015). In these models, a sentence encoder initially provides a dense representation of each sentence regardless of its context in the paragraph. Then, a paragraph encoder creates a context vector to represent the paragraph as a whole. Finally, the decoder, implemented as a pointer network, decides which sentence should be selected next given the paragraph representation as well as the previously selected sentences. A variety of architectures have been explored to use as the encoders and decoders. For example, (Gong et al., 2016) and (Logeswaran et al., 2018) used LSTMs (Zhou et al., 2016) for both the sentence and paragraph encoders as well as for the decoder. On the other hand, given that there is no information regarding the correct order of input sentences and RNN-based models are sensitive to the order of input sequences, (Cui et al., 2018) proposed the use of transformer (Vaswani et al., 2017) as the paragraph encoder. Hierarchical Attention Networks (Wang & Wan, 2019) used an LSTM network for encoding words and transformer encoders for encoding sentences and paragraphs. They also utilized a transformer decoder instead of an LSTM for the pointer decoder.

Some studies tried to take advantage of linguistic features to augment pointer network based models. In particular, Graph-based models (Yin et al., 2019) use a graph encoder to encode a paragraph, where the graph is derived from the linguistic relations between sentences. On the other hand, Topic-guided models (Oh et al., 2019) augment sentence representations with their topics and let the pointer network choose the next sentence based on both the content of the sentences as well as their topics.

Apart from pointer based models, Ranking networks (Kumar et al., 2020) take advantage of the pre-trained BERT model and transformer encoders for encoding sentences and paragraphs. However, as a decoder, they use a feed-forward network to predict a relative score of each sentence as an indicator for their position in the paragraph.

Our proposed model differs from the previous approaches of (Cui et al., 2018; Logeswaran et al., 2018; Wang & Wan, 2019; Yin et al., 2019) and (Oh et al., 2019), as these calculate sentence representations prior to the decoding step, which leads to representations that are independent from the relative position of the candidate sentence. In contrast, our conditional sentence representation is a function of both the content of the candidate sentence as well as the previously selected sentences. In this way, we dynamically calculate a new set of sentence representations at each decoding step where these representations capture the relation between the candidate sentences and the previously selected sentences. Moreover, our model does not use any linguistic features which makes it easy to apply to new datasets or domains.

5.3 Model Description

Our model uses a hierarchical encoder and a decoder module. The encoder module consists of three sub-modules: word encoder, sentence encoder, and paragraph encoder which generate embedding representations corresponding to each word, sentence and paragraph respectively. The decoder, a pointer-based module, receives the output of the word encoder (word embeddings), the sentence encoder (paragraph-aware sentence embeddings), and the paragraph encoder (paragraph embedding) as its input, then, at each decoding step, finds the most probable sentence that should appear next.

In this section we will first define the task, then describe each sub-module in detail.

5.3.1 Task Description

The goal of sentence ordering is to find the most coherent order of a given set S of n sentences that belong to a paragraph P, where $S = \{s_1, s_2, \cdots, s_n\}$, each sentence $s_i = \{w_{i_1}, w_{i_2}, \ldots, w_{i_{L_i}}\}$, L_i is the length of the i^{th} sentence and w_{i_j} is the j^{th} word in s_i. The task is to train a probabilistic model to compute the probability of an ordering $o = \{s_{o_1}, s_{o_2}, \ldots, s_{o_n}\}$:

$$P(o|S) = \prod_{j=1}^{n} P(s_{o_j}|s_{o_{j-1}}, \ldots, s_{o_1}) \tag{2}$$

Given the correct order $o^* = \{s_{o_1^*}, s_{o_2^*}, \ldots, s_{o_n^*}\}$, the objective is to maximize the probability $P(o^*|S)$ among all possible permutations:

$$P(o^*|S) \geq P(o_i|S), \forall o_i \in \psi \tag{3}$$

where ψ is the set of all possible permutations of the sentences s_i in S.

5.3.2 Word Encoder

In our experiments, we use the pre-trained BERT language model (Devlin et al., 2019) as the word encoder module. However, any language model that provides contextual word representations can be used as the word encoder module. The BERT model which uses a transformer encoder (Vaswani et al., 2017), encodes each word w_{ij} in sentence s_i and generates a contextual embeddings $e_{w_{ij}}$ that contains information of the word itself as well as its context (i.e. the sentence). In the following sections, we will describe the main characteristics of the transformer encoder used in our model.

5.3.3 Sentence Encoder

Our sentence encoder consists of two sub-modules: a) a multi-head attention that provides a summary of the word embeddings of each sentence, b) a transformer encoder (Vaswani et al., 2017) that captures the logical relations among sentences.

First, the sentence encoder module receives all the word embeddings $e_{w_{ij}}$ for each sentence s_i and generates the sentence embedding e_{s_i} using a multi-head attention (Vaswani et al., 2017) module followed by a layer norm (LN) (see Eq. 4).

$$
\begin{aligned}
&Attn(Q,K,V) = softmax\left(\frac{QK^\top}{\sqrt{d/H}}\right)V \\
&h_j = Attn(QW_j^Q, KW_j^K, VW_j^V) \\
&MultiHead(Q,K,V) = [h_1; h_2; \ldots; h_H]W^O \\
&MH = MultiHead(Q,K,V) \\
&e_{s_i} = LN(MH + FFN(MH))
\end{aligned} \tag{4}
$$

where W_j^Q,W_j^K, and W_j^V are the weights for the query, key and value of j^{th} attention head (h_j), $MultiHead(Q,K,V)$ is the concatenation of all heads multiplied by the output weights W^O, MH is the output of the multi-head attention, and FFN is a position-wise feed-forward layer. Here we considered word representations $e_{w_{ij}}$ as keys (K) and values (V). We used a fixed query Q (a vector of 1s) and let the model learn which words in the sentence it should give more attention to, in order to get a better sentence representation.

Sentence embeddings e_{s_i} have the semantic representation of the sentences s_i. In order to have a high level representation and capture the logical relations among sentences, following (Cui et al., 2018), we apply a transformer encoder module on the sentence embeddings (e_{s_i}).

Our transformer encoder consists of l identical layers of self attention where the queries, keys and values of the first layer, are all derived from the sentence embeddings (e_{s_i}). Then, the output of each layer (E_{out}^j) feeds as input (E_{in}^{j+1}) to the next layer (see Eq. 5). The output of the last layer contains contextual information of each sentence with respect to the other sentences in the paragraph; hence, they capture the logical dependencies between all sentences. We call these representations:

paragraph-aware sentence embeddings $e^P_{s_i}$.

$$\begin{aligned}
&E^1_{in} = \{e_{s_1}, e_{s_2}, \dots, e_{s_n}\} \\
&E^j_{in} = E^{j-1}_{out} \\
&MH = MultiHead(E^j_{in}, E^j_{in}, E^j_{in}) \\
&E^j_{out} = LN(MH + FFN(MH)) \\
&e^P_{s_i} = E^l_{out}
\end{aligned} \tag{5}$$

5.3.4 Paragraph Encoder

In order to have a paragraph representation (e_P) of the paragraph P, similarly to the previous step (the sentence encoder module), we use a multi-head attention (see Eq. 6) where the query is a fixed vector of 1s and the keys and values are the paragraph-aware sentence embeddings ($e^P_{s_i}$).

$$\begin{aligned}
&MH = MultiHead(Q = [1 \dots 1], K = e^P_{s_i}, V = e^P_{s_i}) \\
&e_P = LN(MH + FFN(MH))
\end{aligned} \tag{6}$$

5.3.5 Decoder

Our decoder is based on pointer networks (Vinyals, Fortunato, & Jaitly, 2015) which consists of an LSTM network augmented with an attention mechanism. At each decoding step, the attention mechanism (pointer) chooses which sentence should be selected among the remaining sentences. The selected sentence passes to the output and its corresponding representation is fed to the LSTM. For the pointer, we use a feed forward attention mechanism using Eq. 7.

$$\begin{aligned}
&q = ReLU(W^q(query)) \\
&k = ReLU(W^k(states)) \\
&scores = W^e(Tanh(q; k)) \\
&pointers = Softmax(scores)
\end{aligned} \tag{7}$$

where $query$ is the hidden state of the LSTM and $states$ are the sentence representations.

In previous work (e.g. (Cui et al., 2018; Logeswaran et al., 2018)), a fixed set of sentence representations (e.g. $e^P_{s_i}$) is used for all the decoding steps. We argue that pointer networks should consider different aspects of the sentences based on previously selected sentences. Hence, we need to dynamically create a new set of sentence representations ($E^t_{s_i}$) at each decoding step t. In our proposed architecture, the pointer attends over these new representations and then we feed the corresponding paragraph-aware sentence representation ($e^P_{s_i}$) to the LSTM.

Figure 5.1 shows the general architecture of our decoder. As illustrated, the decoder receives paragraph embeddings (e_P) from the paragraph encoder module, paragraph-aware sentence embeddings ($e^P_{s_i}$) from the sentence encoder module, and word embeddings ($e_{w_{ij}}$) from the word encoder module. The paragraph embedding (e_P) is fed as the initial hidden state of the LSTM. As the start sentence, a vector of 0s is fed to the LSTM as its input.

In order to create representations that can capture the meaning of a sentences and at the same time capture its relation to the previous sentences, we propose a conditional module. Our conditional module is a multi-head attention (see Eq. 8) where similarly to the sentence encoder, keys (K) and values (V) are the word representations $e_{w_{ij}}$; however, we feed the hidden state (h^{t-1}) of the LSTM network as the query (Q). Since the query Q has the information of previously selected sentences, the attention heads focus more on the information that can connect the sentence to the previously selected sentences. Considering that we use the same query for both the conditional module and the pointer, the representations achieved by the conditional module are aware of what information the pointer is looking for, hence, the conditional module highlights those information in the sentence representations.

$$\begin{aligned} MH &= MultiHead(Q = h^{t-1}, K = e_{w_{ij}}, V = e_{w_{ij}}) \\ C^t_{s_i} &= LN(MH + FFN(MH)) \end{aligned} \tag{8}$$

We can either pass the conditional representations ($C^t_{s_i}$) directly to the pointer or we can pass both the conditional representations ($C^t_{s_i}$) and the paragraph-aware representations ($e^P_{s_i}$) and let the

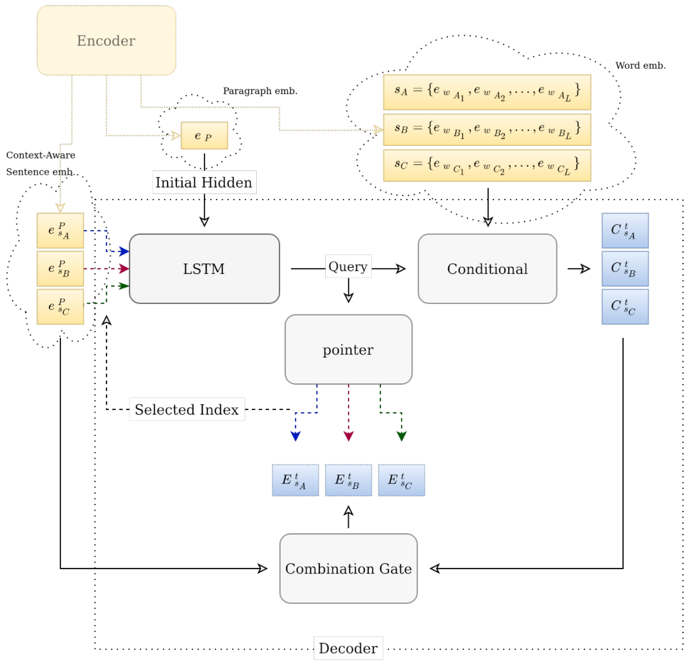


Figure 5.1: General architecture of the pointer decoder

model learn how to use these two representations.

In the case where only the conditional representations are used to choose the next sentence, these representations are passed directly to the pointer mechanism (see Eq. 9). We call this model *Conditional-Only* in Section 5.4.

$$E^t_{s_i} = C^t_{s_i} \tag{9}$$

In the case where these two representations ($C^t_{s_i}$ and $e^P_{s_i}$) are combined, two types of combination are used (see Fig 5.1): via an additive gate and via concatenation. Combining these two representations allows the model to consider both the local dependencies among consecutive sentences ($C^t_{s_i}$) and the global logical relations among all sentences ($e^P_{s_i}$) to choose the next sentence.

Additive gate We calculate the weighted sum of $C^t_{s_i}$ and $e^P_{s_i}$ using a gate g (see Eq. 10). To compute the gate g, first, we feed the concatenation of the two vectors $C^t_{s_i}$ and $e^P_{s_i}$ to a dense layer then a sigmoid function calculates the value of g between 0 and 1. The gate g determines the impact

Dataset	train	valid	test
NIPS	2427	408	377
NSF	87365	8468	21031
SIND	40155	4990	5055
ROC	78529	9816	9817

Table 5.1: Number of sentences in each dataset

of each of $C^t_{s_i}$ and $e^P_{s_i}$ in the final representation.

$$\begin{aligned} g &= \sigma(W_g[C^t_{s_i}; e^P_{s_i}]) \\ E^t_{s_i} &= LN(g \times C^t_{s_i} + (1-g) \times e^P_{s_i}) \end{aligned} \tag{10}$$

Concatenation Another strategy is to simply map the concatenated vector to $E^t_{s_i}$ using a matrix W_s (see Eq. 11).

$$E^t_{s_i} = W_s[C^t_{s_i}; e^P_{s_i}] \tag{11}$$

5.4 Experiments and Results

5.4.1 Datasets

We evaluated our proposed model on two different domains: scientific abstracts and short stories because they have different characteristics. Scientific abstracts are usually well-written and have a clear and stereotypical discourse structure which make them suitable candidates for the task of sentence ordering. On the other hand, short stories usually do not follow a specific discourse pattern which requires the model to capture the intrinsic meaning of the story.

Scientific Abstracts Two datasets of scientific abstracts were used: NIPS (Yin et al., 2019), and NSF (Dua & Graff, 2017). Table 5.1 shows statistics of these datasets.

NIPS Following previous work (Logeswaran et al., 2018), we considered abstract of papers published in NIPS from 2005 to 2013 for training, and 2014 and 2015 for validation and test respectively. We used the parsed dataset provided by (Yin et al., 2019) in their Github repository without any modification. It contains abstracts ranging from 2 to 15 sentences in length.

NSF We also considered the NSF Research Award Abstracts dataset from the UCI Machine Learning Repository (Dua & Graff, 2017) for our experiments. We used abstracts from 1990 to 1999 for training, 2000 for validation and 2001 to 2003 for testing. Due to lack of computational resources, for training and validation, we removed abstracts longer than 15 sentences; however we used abstracts with a maximum of 40 sentences for testing to have a fair comparison with previous work (Logeswaran et al., 2018).

Short Stories For short stories, we used two datasets: SIND (Huang et al., 2016) and ROC (Mostafazadeh et al., 2016) (see Table 5.1).

SIND The SIND (Sequential Image Narrative) dataset (Huang et al., 2016), also known as the VIST (Visual Story Telling) dataset, contains sequential images and their corresponding description and stories. We used the story texts for the task of sentence ordering. Each story has 5 sentences. We used the same training, validation and test sets as provided.

ROC The ROCStory dataset (common sense story dataset) (Mostafazadeh et al., 2016) contains stories of 5 sentences. Following previous work (Kumar et al., 2020; Wang & Wan, 2019), we randomly split the dataset by $8:1:1$ to get the training, validation and testing datasets. Compared to the SIND dataset, the stories contain more explicit signals of discourse coherence since no images are provided as supplementary information.

5.4.2 Experiments

Training setup

We implemented our model using the PyTorch library and trained it using the AdamW (Loshchilov & Hutter, 2019) optimizer with the initial learning rate of $1e^{-4}$ for all modules, except for the word encoder. We did not train the word encoder module for the first two epochs, then, we started training it with a learning rate of $1e^{-5}$. The learning rate was divided by 2 if there was no improvement on the validation set. The training was stopped if no improvement was observed on the validation set in a period of 3 epochs. At the end, the model with the highest performance on the validation set was

Hyper-Parameter	**NIPS**	**NSF**	**SIND**	**ROC**
# layers	2	4	2	4
Model dim	256	128	64	128
LSTM dim	128	256	128	512

Table 5.2: Hyper-parameters for each model. # layers indicates the number of attentions layers. Model dim refers to the embedding size of the model. LSTM dim is the hidden size of LSTM in the pointer network.

Model	**Scientific Abstracts**						**Short Stories**			
	NIPS			**NSF**			**SIND**		**ROC**	
	Acc	PMR	τ	Acc	PMR	τ	PMR	τ	PMR	τ
LSTM+PtrNet	51.55	–	0.72	28.33	–	0.51	–	–	–	–
ATTOrderNet	56.09	–	0.72	37.72	–	0.55	14.01	0.49	–	–
Hierarchical	–	–	–	–	–	–	15.01	0.5021	39.62	0.7322
Ranking	–	24.13	0.7462	–	9.78	0.5798	15.48	0.5652	38.02	0.7602
SE-Graph	57.27	–	0.75	–	–	–	16.22	0.52	–	–
Topic-Guided	59.43	31.44	0.75	**42.67**	**22.35**	0.55	15.18	0.53	–	–
Baseline	64.71	29.17	0.7962	38.66	10.68	0.5818	15.66	0.5607	36.79	0.7554
Add-Conditional	64.97	30.76	0.7950	41.86	13.55	**0.6060**	**17.46**	0.5640	39.53	0.7652
Cat-Conditional	**65.73**	**31.83**	**0.7990**	41.46	13.19	0.6029	17.01	0.5592	39.46	0.7660
Conditional-Only	63.42	30.23	0.7843	41.56	13.44	0.6006	16.99	**0.5672**	**42.01**	**0.7726**

Table 5.3: Experimental results of different methods on standard sentence ordering datasets

chosen for testing. The weights of the word encoder module were initialized with the pre-trained BERT base model from HuggingFace (Wolf et al., 2019) with 12 encoder layers. The following hyper-parameters were chosen based on the validation sets without extensive hyper-parameter optimization: we used 4 heads of attention with a dropout rate of 0.2 for all multi-head attention modules. We also considered dropout rates of 0.5 and 0.3 for the word encoder and pointer respectively. The rest of the hyper-parameters, which were dataset-dependent are shown in Table 5.2.

Models We experimented with three models that use our proposed conditional sentence representation with three different strategies described in Section 5.3.5. All these models use a similar encoder and feed the same sentence representation to the LSTM; the only difference between these three models is in the way they use the conditional representations (i.e. the combination of $C^t_{s_i}$ and $e^P_{s_i}$):

(1) *Add-Conditional* uses the additive gate in the decoder to combine both representations ($C^t_{s_i}$) and ($e^P_{s_i}$) using ($g \times C^t_{s_i} + (1-g) \times e^P_{s_i}$), where g is learned by the model.

(2) *Cat-Conditional* uses a decoder that concatenates both representations ($[C^t_{s_i}; e^P_{s_i}]$).

(3) *Conditional-Only* only passes the conditional representations ($C^t_{s_i}$). This model is equivalent to *Add-Conditional* with $g = 1$.

In order to better understand how the conditional sentence representations impact the performance, we also considered a baseline model without using the conditional sentence representation:

(4) *Baseline* only passes the paragraph-aware representations ($e^P_{s_i}$). This model is equivalent to *Add-Conditional* with $g = 0$.

The performance of these four models on the benchmark datasets of Section 5.4.1 are shown in Table 5.3, along with the performance of state-of-the-art models as reported in their paper. Recall from Section 5.2 that LSTM+PtrNet (Logeswaran et al., 2018) used LSTMs for both the sentence and paragraph encoders with pointer networks. ATTOrderNet (Cui et al., 2018) substituted the LSTM for a paragraph encoder with a transformer mechanism. Hierarchical attention (Wang & Wan, 2019) used transformer for both the paragraph encoder and the pointer. Ranking Model (Kumar et al., 2020) used a regression approach to predict a relative score for each sentences as an indicator for their positions. SE-Graph (Yin et al., 2019) augmented the pointer network with a graph neural network to encode sentences. Topic-Guided (Oh et al., 2019) added sentences representation with their topics to have more information for each sentences.

Evaluation Metrics Accuracy (Acc) or position-wise accuracy measures the absolute position of each sentence. It penalizes shifts even with a correct relative order.

Perfect Match Rate (PMR) is a hard metric that counts the number of exact matching paragraphs over the total number of paragraphs. It is hard because if a single sentence is mis-ordered, it considers the entire paragraph as wrong.

Kendall's Tau (τ) is a rank correlation metric which measures how the predicted order is similar to the given order. The formula of Kendall's Tau is given in Eq. 12 where the number of inversions is the number of pairs with incorrect relative order and n is the length of the paragraph. Its range is between -1 to 1, from worst to best.

$$\tau = 1 - \frac{2(\#inversion)}{n(n-1)/2} \tag{12}$$

5.4.3 Results and Analysis

Table 5.3 reports on the experimental results on all datasets. As the table shows, all models with the conditional sentence representations ($C^t_{s_i}$) achieved a better performance than the *Baseline* model.

As expected, models that used our proposed representations achieved a significantly better performance on the short stories datasets (SIND and ROC) compared to *Baseline* (the model using paragraph-aware sentence representations $e^P_{s_i}$ only). In these short stories, local dependencies and the relation between consecutive sentences have a significant impact on the order of sentences. Hence, we can inferred from these results that the conditional sentence representations ($C^t_{s_i}$) were able to capture these local dependencies successfully.

On the NIPS dataset, the *Cat-Conditional* model achieved better results than the *Baseline* model, but this time, the improvement is not statistically significant. On the other hand, *Conditional-Only* achieved the lowest performance among the models with conditional representations. This seems to agree with the fact that in these abstracts, sentences follow a clear discourse structure and local dependencies among consecutive sentences have a lower impact on the correct order of sentences compared to other datasets such as short stories.

Results with the NSF dataset indicate that all models that used the conditional sentence representations achieved similar performances ($\tau \approx 0.60$); while the model without this representation suffers from a significant drop in performance ($\tau \approx 0.58$). This behavior does not occur with the NIPS dataset, where the performance with and without the conditional representation does not show such a marked difference. This is rather surprising as both datasets are scientific abstracts and hence should share a similar discourse structure. We suspect that this behaviour stems from the strict writing guidelines imposed by NSF, which aim to simplify the language, the grammar and the structure of the abstracts, that the conditional representation picks up on. However, more analysis is required to investigate this.

Compared with previous work (Cui et al., 2018; Kumar et al., 2020; Wang & Wan, 2019), the models with conditional sentence representations ($C^t_{s_i}$) achieved better results on most metrics on all datasets. Results show that the *Baseline* model (which does not use $C^t_{s_i}$) achieved similar or lower results than previous work on most datasets. This clearly shows that our highest performance is not related to the settings we used for our training, but rather the use of the conditional representations ($C^t_{s_i}$) alone. For the NIPS dataset, we cannot easily compare our results with previous work since the number of samples in our test sets is different from the one reported in previous work (Logeswaran et al., 2018). In addition, since the NIPS dataset is much smaller than the other datasets (see Table 5.1), using a pretrained model such as BERT has a significant impact on its performance. However, the Ranking model of (Kumar et al., 2020) also used the same pretrained model yet its performance lies in the same range as other work (Oh et al., 2019; Wang & Wan, 2019).

In order to better understand the impact of the conditional representations, we analyzed the focus of the attention heads. Figure 5.2 illustrates the values of all the attention heads of the sentence encoder and the conditional modules of the *Add-Conditional* model trained on the ROC dataset alongside with their corresponding value of g gate. In Figure 5.2, the conditional modules are shown on the left hand side; while the heads of the sentence encoder are shown on the right hand side. Recall that the output of the sentence encoder is independent of the position of the sentences, while this is not the case for the conditional module. Figure 5.2 shows the heads of the conditional module only for the correct position of each sentence. As Figure 5.2 shows, the sentence encoder module tends to attend to all tokens in the sentence to capture the meaning of the sentence; however, the conditional module focuses on specific tokens based on the relation of the sentence to the previous sentences. For example, in the first sentence *"Judy wanted to make some cookies."*, since there is no previously selected sentence, more attention is given to the subject of the sentence (*Judy*); however, on the third sentence, the attention focuses on the specific words *without* and *it*, where *it* refers to *butter* in the previous sentence, another head attends more on *decided* and *to* which connects the sentence to the previous one.

We further analyzed the value of the g gate to see how the two sentence representations contribute in the final representation. Table 5.4 shows the average value of g for sentences in their correct position. As Table 5.4 indicates, for the first sentences, where there is no previously selected sentences therefore, the paragraph-aware representations ($e^P_{s_i}$) contribute more to the final representations ($E^t_{s_i}$). However, for the rest of the sentences, g is closer to 0.5 leading to a more equal contribution of the conditional representation ($C^t_{s_i}$) and the paragraph-aware representations ($e^P_{s_i}$). Again, this agrees with our intuition that the conditional representation tends to capture local dependencies between consecutive sentences and the gate g puts more weight on the conditional representation when information from previously selected sentences is available.

Sentence	g
1	0.3 ± 0.1
2	0.5 ± 0.1
3	0.6 ± 0.1
4	0.6 ± 0.1
5	0.6 ± 0.1

Table 5.4: The value of the gate g in the *Add-Conditional* model trained on the ROC dataset. Recall from Eq. 10 that a higher value of g indicates more contribution of the conditional representation ($C_{s_i}^t$).

5.5 Conclusion

In this chapter, we proposed a model based on pointer networks for the task of sentence ordering. The model uses a hierarchical encoder to encode words, sentences and the paragraph as a whole. Then the decoder uses all these embeddings to predict the next sentence in the paragraph. Our decoder considers a novel conditional sentence representation, which combines both the content of the sentence and the previously selected sentences in the text. Experiments with standard benchmark datasets show that the proposed model achieves state-of-the-art results using most of the evaluation metrics. The increase in performance is more significant in short stories datasets, where local dependencies in consecutive sentences play an important role in the discourse structure of the paragraph.

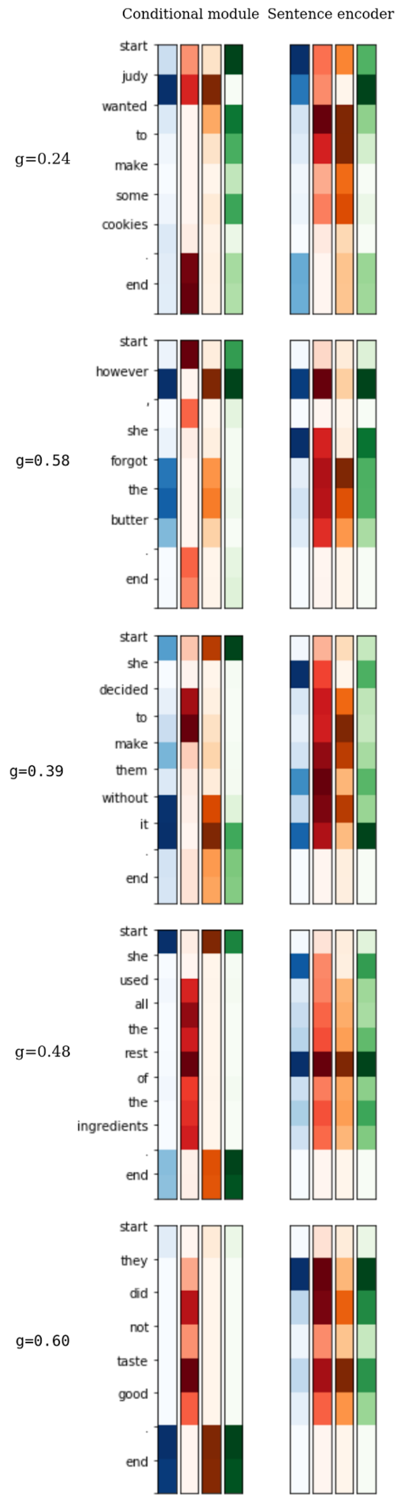


Figure 5.2: Visualisation of the attention heads for the *Add-Conditional* model trained on the ROC dataset with their corresponding values of the g gate. The heads of the conditional module are on the left and the heads of the sentence encoder module are on the right.

Chapter 6

Pre-training Language Models for Surface Realization

In Chapter 4, we showed that a text-to-text Language Model can address the Surface Realization problem and achieve competitive performance to the state-of-the-art. In this chapter, we try to improve the performance of this model by pre-training it on the synthetic training data and then fine-tuning it on the manually labeled data. In this chapter we try to answer the forth research question formulated in Section 1.3: “Can we improve the performance of surface realizer models with less amount labeled training data than the current the state-of-the-art models?”.

Please note that the content of this chapter has been published at the 5th International Conference on Natural Language and Speech Processing, ICNLSP-2022 (Farahnak & Kosseim, 2022).

Abstract

Surface Realization in Natural Language Generation (NLG) is the task of deriving the surface form of a sentence (the actual words) from an underlying representation. Following recent advances in deep learning, several models have been proposed for different NLG sub-tasks including surface realization. Most of these models require a large amount of training data, however, acquiring accurately labeled data is laborious and expensive. In this work, we study how synthetically generated labeled data can be leveraged to improve the performance of a surface realization model.

By pre-training a language model on automatically labeled data and then fine-tuning it on manually labeled data, our approach improved the state-of-the-art performance on the standard English datasets from the deep track of the Multilingual Surface Realization (MSR) workshop (Belz et al., 2020) by more than 10% BLEU score.[1]

6.1 Introduction

The goal of Natural Language Generation (NLG) is to generate text in human languages (e.g. English) for a wide range of applications such as report generation, text summarization, and conversation modeling. NLG involves both *content planning* (selecting the content to communicate) and *surface realization*. Surface realization (SR), the last step of the NLG pipeline, aims to derive the surface form of a sentence (the actual words) from an underlying representation by choosing the proper word forms (inflection, punctuation, and formatting) and determining their correct order (syntactic realization) (Hovy et al., 1996; Reiter & Dale, 2000).

Recent advances in Natural Language Processing (NLP) and Deep Neural Networks (DNN) have led to drastic improvements in many NLP systems, some of which have even achieved human-level performance (Läubli, Sennrich, & Volk, 2018). Similarly to many NLP models, surface realization models have also benefited from these advancements. DNN models usually require a large amount of labeled data for training; however, creating accurate and reliable training data is an expensive and time consuming task. In this work, we show how we can improve the performance of surface realization by pre-training a language model on a large synthetically generated dataset and then fine-tuning it on a smaller manually labeled dataset.
To measure the effectiveness of our approach, we followed the protocol of the Multilingual Surface Realization (MSR) Workshops (Belz et al., 2020; Mille et al., 2019, 2018), and generated the surface form of sentences from their dependency parse trees. To create the synthetic data, we used the automatic dependency parser Stanza (Qi et al., 2020) to parse the unlabeled WikiText corpus (Merity et al., 2017). Using different sizes of manually labeled and synthetic data, we investigated the effects of the proposed pre-training phase. Although the synthetic data may contain noisy annotations

[1]The code is available at `https://github.com/CLaC-Lab/SR_LM`

compared to manually labeled data and may come from a different distribution (e.g. different textual genre or discourse domain), results show that its sheer size allows the model to learn the general gist of the task in the pre-training phase and leads to an increase in performance in SR achieving state-of-the-art performance on the deep track with the English datasets of the MSR workshops.

6.2 Related Work

6.2.1 Multilingual Surface Realization (MSR)

The Multilingual Surface Realization (MSR) workshops have organized shared tasks aimed at bringing together researchers interested in surface oriented Natural Language Generation problems and share resources to that end (Belz et al., 2020; Mille et al., 2019, 2018). The shared task aimed to generate the surface form of sentences given their Universal Dependency (UD) structures. Two tracks were proposed: the shallow and the deep tracks. For the shallow track, word order information and the inflected form of words were removed from the UD structure and the task aimed to determine the correct order of words and inflect them. In the deep track, in addition to word ordering and inflection, functional words (in particular, auxiliaries, functional prepositions and conjunctions) and surface-oriented morphological information were removed from the UD structure and had to be recovered by the models.

6.2.2 Previous Work

Participants in the Multilingual Surface Realization (MSR) workshops proposed different models to address the surface realization task. Many of these models use dedicated sub-modules for each sub-task. For example the ADAPT center (Elder, 2020) proposed a biLSTM sequence-to-sequence model with a copy mechanism to generate the surface form of sentences. They augmented the training set with 4.5M sentences from two sources, WikiText (Merity et al., 2017) and CNN stories (Hermann et al., 2015), and chose sentences that had at least 80% word overlap with the labeled dataset to ensure that they have a similar distribution. The BME-TUW system (Recski, Kovács, Gémes, Ács, & Kornai, 2020) used an Interpreted Regular Tree Grammar to retrieve the correct order of tokens then used a biLSTM sequence-to-sequence model to inflect the words. The IMS system (Yu, Tannert, Vu, & Kuhn, 2020) tackled the surface realization problem as a Traveling Salesperson Problem, and used a biaffine attention model to calculate the bigram scores for the output sequence. Finally, they used a biLSTM for the inflection module. Similarly to the ADAPT center, IMS also used WikiText and CNN stories to augment their training data with $200K$ synthetic samples, however, by considering the branching factors of the tree, they tried to keep the distribution of the augmented data close to the labeled datasets. The data augmentation that ADAPT and IMS used differ from our proposed solution as they both tried to keep the distribution of the augmented data as similar as possible to the manually labeled data by applying filtering rules. In contrast, our approach does not enforce the distributions to be similar, and lets the domain adaptation to be performed automatically during the fine-tuning phase.

Because of their simplicity and effectiveness, several approaches have used language models for surface realization. The NILC system (Cabezudo & Pardo, 2020) proposed to use GPT-2 (Radford et al., 2019) and linearization using the parentheses approach. We argue that when the number of nodes grows, the model has difficulties in capturing the relations between them. The Concordia system (Farahnak et al., 2020) used BART (Lewis et al., 2020) for surface realization, however, the relation between nodes was represented with the actual words. This approach may cause problems when a word appears more than once in a sentence as the model cannot capture the exact structure of the tree. Our approach is also based on language models, however, it differs from theirs as indices are used to encode the edges in the UD structure instead of the actual tokens (see Section 6.4).

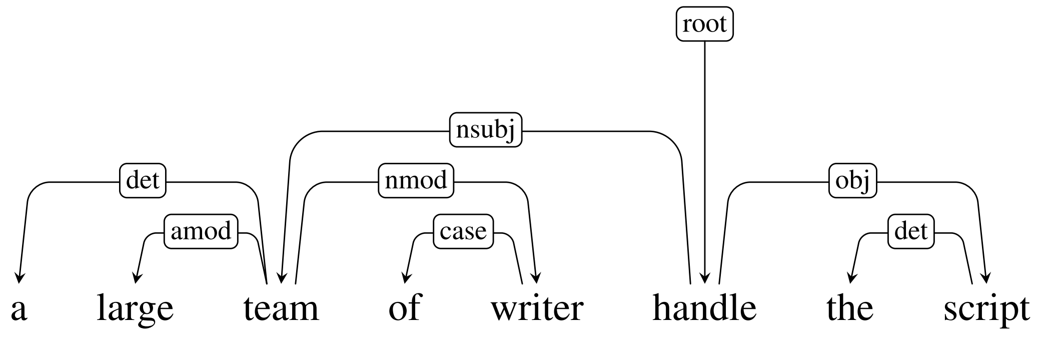


Figure 6.1: Example of UD dependency parse tree for the sentence *A large team of writers handled the script*.

6.3 Data

In this section, we first present the MSR manually labeled datasets we used for our experiments and then discuss how we created the synthetic dataset.

6.3.1 Manually Labeled Datasets

For our experiments, we used the English datasets provided by the MSR workshop (Belz et al., 2020). These datasets are modified versions of the Universal Dependency (UD) datasets (de Marneffe et al., 2014) where the order of the tokens is shuffled and the inflected form of the tokens are removed. Table 6.1 presents statistics of these datasets. As Table 6.1 shows, the largest dataset (EWT) contains only $\approx 12K$ training samples which makes it hard to train an DNN model based solely on these samples.

Dataset	train	dev	test
EWT	12,543	2,002	2,077
GUM	2,914	707	778
LinES	2,738	912	914
ParTUT	1,781	156	153

Table 6.1: Number of samples in the English MSR datasets.

6.3.2 Synthetically Generated Dataset

In order to generate synthetic data, we used the WikiText dataset (Merity et al., 2017), extracted from Wikipedia articles. The WikiText dataset comes from a different domain compared to the MSR datasets[2] which makes it a suitable candidate to study the domain adaptation between the two text genres. We extracted the first $500K$ sentences after filtering non-English sentences and sentences longer than 150 characters[3] to create our synthetic dataset. We used Stanza (Qi et al., 2020) to parse the sentences and create their UD structure. Using the script provided by the MSR workshop (Belz et al., 2020), we generated the synthetic dataset in the same format as provided by the workshop.

[2]The EWT dataset contains sentences from five genres of web media: weblogs, newsgroups, emails, reviews, and Yahoo! answers.

[3]This value was chosen because 90% of the samples in EWT are shorter than 150 characters.

Figure 6.1 shows a visual representation of the dependency tree structure of a sample from the dataset.

# synthetic samples for pre-training	Shallow Track								Deep Track							
	EWT	Δ	GUM	Δ	LinES	Δ	ParTUT	Δ	EWT	Δ	GUM	Δ	LinES	Δ	ParTUT	Δ
0	80.79	_	71.63	_	69.62	_	67.84	_	64.31	_	48.74	_	40.61	_	49.23	_
$100K$	84.38	3.59	86.27	14.64	82.38	12.76	86.98	19.14	68.10	3.79	68.61	19.87	64.28	23.67	69.50	20.27
$200K$	84.62	3.83	86.84	15.21	83.00	13.38	86.69	18.85	69.02	4.71	69.21	20.47	65.29	24.65	69.25	20.02
$500K$	**84.69**	3.90	**86.76**	15.13	**83.18**	13.56	87.66	19.82	**69.52**	5.21	**70.19**	21.45	**66.26**	25.65	**71.38**	22.15

Table 6.2: BLEU score of models pre-trained with different sizes of synthetic data. Δ reports the difference of the pre-trained models to training without the pre-training phase (i.e. 0 synthetic data).

		EWT			GUM			LinES			ParTUT		
		BLEU	NIST	DIST	BLEU	NIST	DIST	BLEU	NIST	DIST	BLEU	NIST	DIST
Shallow Track	BME (Recski et al., 2020)	57.25	12.52	65.23	60.77	12.10	62.86	55.98	11.78	61.44	61.37	10.22	58.39
	Concordia (Farahnak et al., 2020)	70.71	12.70	77.94	66.98	11.62	69.87	62.70	11.30	68.62	67.05	9.83	71.59
	IMS (Yu et al., 2020)	85.67	13.74	87.74	**89.70**	**12.98**	**91.97**	**85.30**	**12.97**	**86.48**	**89.37**	**11.05**	**88.73**
	ADAPT (Elder, 2020)	**87.50**	**13.81**	**90.35**	_	_	_	_	_	_	_	_	_
	Our Approach (Farahnak & Kosseim, 2022)	84.69	13.58	88.82	86.76	12.65	89.12	83.18	12.59	85.72	87.66	10.91	86.80
Deep Track	NILC (Cabezudo & Pardo, 2020)	45.19	9.96	64.83	53.92	9.00	60.42	41.04	9.09	61.18	43.41	8.24	59.74
	Concordia (Farahnak et al., 2020)	58.44	11.61	73.66	53.92	10.51	67.02	47.96	9.93	64.33	50.54	8.57	62.39
	IMS (Yu et al., 2020)	58.66	11.61	79.23	53.92	11.25	76.47	50.45	10.89	73.1	50.11	9.26	72.98
	Our Approach (Farahnak & Kosseim, 2022)	**69.52**	**12.54**	**82.43**	**70.19**	**11.64**	**80.93**	**66.26**	**11.37**	**78.81**	**71.38**	**9.99**	**77.88**

Table 6.3: Comparison of our approach (models pre-trained on $500K$ synthetic sentences and fine-tuned on each dataset) with previous models proposed for the deep track of MSR 2020.

6.4 Model

Following the success of pre-trained language models (LMs) for data-to-text generation tasks (Farahnak et al., 2020; Harkous et al., 2020; Kale & Rastogi, 2020), we used an encoder-decoder LM for surface realization. The input and output of an encoder-decoder LM is in linear form (text-to-text); however, surface realization is a data-to-text task. In order to use LM, the input UD structure had to be linearized. Among the features available in the UD structure, we considered `lemma` (the lemmatized form of tokens), `FEATS` (morphological information), `HEAD` (the parent in the tree structure), and `deprel` (dependency relation to the head) and represented each node in the linear format:

```
index : lemma FEATS : head_index <deprel>
```

then concatenated all nodes together. Figure 6.2 shows the linearized representation of the example from Figure 6.1 used for the shallow track. In this example, the parent of the word `script` is node 5 which is the index for word `handle`. To train the LM, we used the surface form of the sentence as the target. The model learns to generate the surface form given the linearized UD structure, hence, it learns to perform both syntactic and morphological realization simultaneously.

```
4 : script Sing : 5 <obj> # 3 : writer Plur : 7 <nmod>
# 9 : . : 5 <punct> # 6 : large Pos : 7 <amod> # 7 :
team Sing : 5 <nsubj> # 5 : handle Ind Plur 3 Past
Fin : ROOT <root> # 1 : the Def Art : 4 <det> # 8 :
of : 3 <case> # 2 : a Ind Art : 7 <det>
```

Figure 6.2: Linearized representation of the UD structure of Figure 6.1.

6.5 Experiments and Results

6.5.1 Experimental Setup

In order to understand the effect of synthetic data on the performance of the ordering model, we conducted several experiments using different sizes of synthetic data to pre-train the model, then fine-tuning it on the manually labeled datasets and measuring the performance on the MSR test sets (see Table 6.1). For all experiments, we used the pre-trained BART (Lewis et al., 2020) large model. We used the AdamW (Loshchilov & Hutter, 2019) optimization algorithm with a learning rate of $1e$-5 and batch size of 4 to train our models. We pre-trained the models for 5 epochs on the synthetic data and fine-tuned them for 5 more epochs on the manually labeled data. For comparative purposes, we also trained the models without the pre-training phase, and trained them for 15 epochs on the manually labeled data. We choose the model with the highest performance on the development sets.

6.5.2 Results

Table 6.2 compares the performance of training the encoder-decoder language model using different sizes of synthetic data for pre-training. Our experiments suggest that the pre-training phase can improve the performance of the model by 3.90% and 5.21% in BLEU score for the shallow and deep tracks respectively on the EWT dataset. However, the improvement on the other three datasets are more significant, ranging from 12.76% to 25.65%, as these datasets have much fewer training samples compared to EWT. The improvement of pre-training on synthetic data is higher for the deep track compared to the shallow track as the task is more complex in the sense that the model not only needs to learn the inflection and ordering of words, it also needs to guess the removed functional words.

In comparison with previous participating models of MSR 2020 (Belz et al., 2020) (see Table 6.3), our approach is not able to outperform the previous work on the shallow track. However, it improves the state-of-the-art performance by a large margin (more than 10% in BLEU score) on in the deep track on all datasets which shows the superiority of our proposed approach.

6.5.3 Analysis

We analysed the results of the models to better understand the benefits and drawbacks of our approach.

Pre-training seems to facilitate domain adaption, as a single epoch of fine-tuning is enough for the model to adapt to the domain of the manually labeled dataset (see Section 6.6.1).

Pre-training can significantly reduce the need for manual data. We fine-tuned the pre-trained models using subsets of the manually labeled data. Results shows that with pre-training, using only 10% of the data achieves better performance than training on all manually labeled data without the pre-training phase (see Section 6.6.2).

Finally, through a manual inspection of the generated sentences (see Section 6.6.3), we determined that most errors should actually be considered correct alternatives to the ground truth. Better automatic measures should be developed to measure the performance of surface realization to account for linguistic variations.

6.6 Detailed Analysis

6.6.1 Domain Adaptation

Figure 6.3 compares the BLEU scores of training models for the deep track on the EWT dataset with and without pre-training on $500K$ synthetic samples with different training epochs. As the figure shows, for the pre-trained model, the domain adaptation phase is almost completed after the first epoch while the non-pre-trained model continues to improve even after 10 epochs.

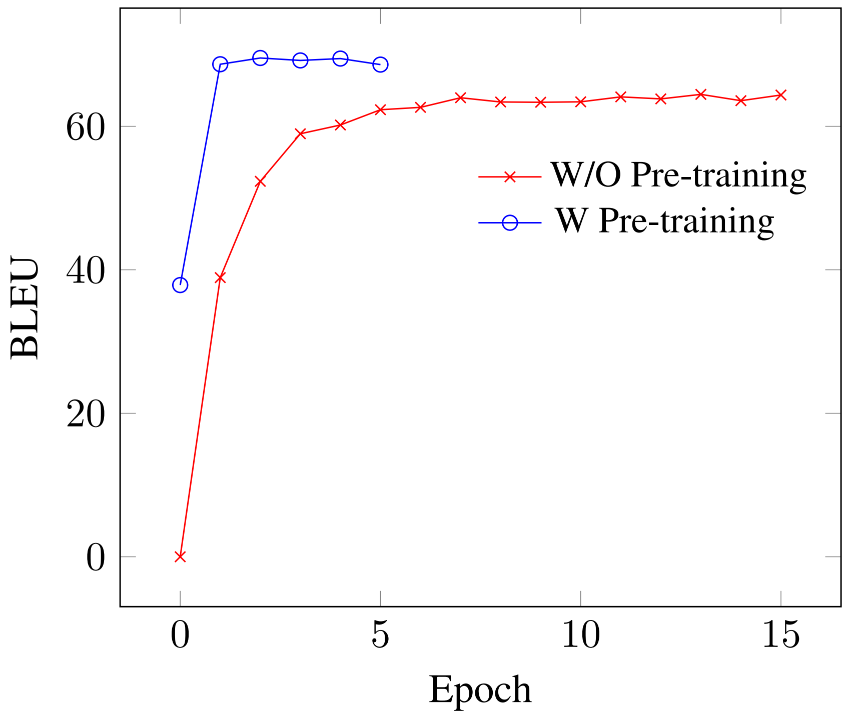


Figure 6.3: BLEU score on the EWT test set for all epochs when training the models for the deep track, with and without pre-training on $500K$ synthetic samples.

6.6.2 Size of Training Data

Table 6.2 shows that pre-training on synthetic data before fine-tuning on manually labeled data can improve the overall performance of the model. In order to better understand the importance of the size of manually labeled training data, we limited its size and fine-tuned the model on different sizes of manually labeled training data. Table 6.4 shows the performances of the model after fine-tuning on different subsets of the EWT dataset. As Table 6.4 shows, fine-tuning solely on $1K$ samples can achieve better performance compared to no pre-training and using the full EWT dataset (last row of Table 6.4). However, by increasing the number of training samples (from $1K$ to $12.5K$), we can achieve a higher performance when pre-training. This indicates that even though the pre-training phase is helpful for the task, it is not sufficient to replace the manually labeled training data altogether.

# synthetic samples	# manually labeled samples	BLEU	NIST	DIST
$500K$	0	37.87	8.09	61.04
$500K$	1K	64.54	11.88	78.50
$500K$	2K	65.70	12.00	78.93
$500K$	5K	67.35	12.33	80.76
$500K$	10K	68.79	12.43	81.80
$500K$	12.5K	69.52	12.54	82.43
0	12.5K	64.31	11.64	77.80

Table 6.4: Comparison of the performance of the encoder-decoder model using different sizes of training data for the fine-tuning on a model pre-trained with $500K$ synthetic samples.

6.6.3 Error Analysis

We manually inspected the errors generated by our models. While a few generated sentences did contained true errors, most can be regarded as correct alternatives to the ground truth. Table 6.5 shows a few examples. One common correct alternative was related to the generation of contractions as in Ex. 1. This type of error occurs because the MSR input structure of the token to generate (*it*) does not contain any feature that give the model a clue as to whether the token should be contracted or not. In Ex. 2, the model failed to generate the expected punctuation in the deep track, yet the generated sentence is a correct alternative to the ground truth. In Ex. 3, the word order in the generated outputs is not identical to the ground truth; however, they are grammatically correct and convey the same meaning. In Ex. 4, the output of the shallow model is indeed a true error as it is not grammatically correct; however, the deep model generated a grammatically correct output but again it is not identical to the expected output. Finally, in Ex. 5 and 6, show examples of correct alternative to number formatting compared to the ground truth.

6.7 Conclusion and Future Work

In this chapter, we showed that pre-training on synthetic data is beneficial for surface realization even when the data comes from a different distribution than the training data. We also showed that the pre-training phase not only improves the performance of the model, but also helps the model to converge faster on the training data. The proposed pre-training phase for LM improved

	Ground Truth	i 'll post highlights . . .		Ground Truth	two weeks later , and the violence continues .
Ex. 1	Output of Shallow	i will post highlights . . .	Ex. 2	Output of Shallow	two weeks later , and the violence continues .
	Output of Deep	i will post highlights . . .		Output of Deep	two weeks later and the violence continues .
	Ground Truth	they own blogger , of course .		Ground Truth	we have this report ?
Ex. 3	Output of Shallow	of course , they own blogger .	Ex. 4	Output of Shallow	have we this report ?
	Output of Deep	of course they own blogger .		Output of Deep	do we have this report ?
	Ground Truth	compensation : $ 60000 - 70000		Ground Truth	. . . said that there was a 10 to 50 % chance . . .
Ex. 5	Output of Shallow	compensation : $ 60,000 - 70,000	Ex. 6	Output of Shallow	. . . said that there was a 10 to 50 % chance . . .
	Output of Deep	compensation : $ 60000 - 70000		Output of Deep	. . . said there was a 10 - 50 % chance . . .

Table 6.5: Sample errors generated by the shallow and deep models pre-trained on $500K$ synthetic data and fine-tuned on the EWT dataset.

the state-of-the-art performance on the standard English datasets from the deep track of the MSR workshop (Belz et al., 2020) by more than 10% BLEU score.

As of future work, we plan to conduct similar experiments on previously proposed models such as ADAPT (Elder, 2020) and IMS (Yu et al., 2020). We also plan to run cross-language experiments to see whether the knowledge learned from one language can be transferred to another language.

Chapter 7

Conclusion and Future Work

In this thesis, we address the question of ordering problems in NLG. We considered two sub-tasks: Surface Realization and Sentence Ordering. In this chapter, we will review our contributions, then highlight the limitations of our work and finally, we will describe how we can further improve our suggested approaches.

7.1 Contributions

In this section, we review the contribution of this thesis. As we discussed in Chapter 1, in order to achieve our goal, we defined the following research questions (see Section 1.3):

- RQ1. Can we address the problem of surface realization as an ordering problem? (See Chapter 3)
- RQ2. Can we use a standard (text-to-text) language model to address the task of surface realization? (See Chapter 4)
- RQ3. Can we design a sentence representation that captures the coherence relations between sentences? (See Chapter 5)
- RQ4. Can we improve the performance of surface realizer models with less amount labeled training data than the current the state-of-the-art models? (See Chapter 6)

We were able to answer these research questions, through the following contributions:

(1) We proposed an approach to Surface Realization that uses transformers and pointer networks to generate coherent texts from an abstract representation. (See Chapter 3)

In this approach, the transformer module encodes the input graph and the pointer network finds the correct order of nodes. Finally a decoder module generates the surface form of tokens. The final approach can be used in other NLG pipelines to generate text.

To the best of our knowledge, this is the first attempt to use transformers and pointer networks for the task of Surface Realization.

(2) We proposed an approach to take advantage of pretrained text-to-text language models (LMs) for the task of Surface Realization. (See Chapter 4)

We represent the input graph in a text format and then feed it to the LM as the input and we use the surface form of the sentence as the target output to train the LM. The LM model learns to perform all intermediate steps (i.e. encoding, ordering, and inflection) at the same time.

To the best of our knowledge, this is the first attempt to use text-to-text LMs for the task of surface realization. The proposed model, achieved a BLEU score of 58% in the deep surface realization track on the `en_ewt-ud` dataset which was competitive with other participants.

(3) We proposed an approach for the task of Sentence Ordering that is able to find the most coherent order of a given set of sentences. (See Chapter 5)

In our approach, in contrast with previous approaches where sentences are encoded individually regardless of their position, we proposed to encode sentences not only based on their content, but also based on their position. This new sentence representation which is conditioned on both the content of the sentence and the content of previously selected sentences is able to better capture the relation between sentences and hence improve the performance of the model. The proposed model can be used for any NLG task that requires generating more than one sentence. Also, this approach can be used for tasks where the coherence of written text should be measured such as essay evaluation. Our approach improve state-of-the-art performance on short stories and achieved a PMR of 17.46% and 42.01% on the SIND and ROC datasets.

(4) We proposed an approach to improve the performance of current surface realization approaches without the need to increase the amount of manually labeled training data. (See Chapter 6)

    In this approach, we take advantage of transfer learning by first training a model using synthetic data and then fine-tuning it on the target dataset. We have created the synthetic dataset by parsing raw texts. Although the created dataset may be noisy, it was shown to help model to learn the task of surface realization. Then, by fine-tuning the manually labeled data, we have adapted the model to the distribution of the target dataset. Using different sizes of manually labeled data to fine-tune the model, we measured the effectiveness of our approach. Our approach achieved a BLEU score of 69.5% on the deep surface on `en_ewt-ud` dataset which was 10% more than the state-of-the-art.

    All the code and necessary resources to run the experiments are available at `https://github.com/CLaC-Lab/SR_LM`.

7.2 Limitations

7.2.1 Sentence Ordering

As Chapter 5 described, we tested our approach on scientific abstracts and short stories. In both these textual genres, the input text was well written and well structured. However, many texts and essays do not follow a single well-defined structure. In an essay, the order of sentences may follow the chronological order of events and recovering this order requires external knowledge to be infused into the model. Our proposed model does not address this. Another limitation of our model is when the input contains unrelated sentences. In this situation, the model should be able to find and sort the related sentences. However, in our approach, the model sorts all sentences and if there is an unrelated sentence, it will degrade the performance of the model. Finally, although the experiments indicate that our proposed model improved the performance by better capturing the relations between sentences, it does not identify such relations. Identifying the coherence relation between sentences were out of the scope of this research. However, having such information can not only improve the performance of the model, but can also help have an explanation behind the predicted order and improve model interpretability (Liu, Wang, & Matwin, 2018).

7.2.2 Surface Realization

For surface realization we followed the setting proposed by the Multilingual Surface Realization (MSR) workshops (Belz et al., 2020; Mille et al., 2019, 2018) in which we use the dependency structure of actual sentences to feed to our model and then compare the output with the gold standard sentence. The first limitation of this approach is regarding the evaluation. We used BLEU and other automated metrics to see how much the generated sentence is close to the expected sentence. When the model generates a correct sentence that is different from the expected sentence, the automated score penalizes the model. Although in the last MSR workshop (Belz et al., 2020), the organisers considered the human evaluation, it is not easy (if possible) to replicate the human evaluation for other experiments hence comparing the performance with previous work would be limited to the automated evaluation. The second limitation of this approach is related to the structure of the input. Although in the deep track the organisers removed functional words and the model needs to guess those words, in a real scenario the input to the model would be more abstract compared to the dependency parse tree.

7.3 Future Work

In this thesis, we introduced conditional sentence representation to capture the relation between sentences. We showed that enhancing pointer networks with conditional sentence representation can further improve the performance of a sentence ordering model. However, the usage of conditional sentence representation is not limited to sentence ordering. In the task of semantic similarity, Farahnak, Mohammadi, Davari, and Kosseim (2021) showed that a conditional representation can improve the performance of the model compared with a static sentence representation. As future work, we would like to extend this approach to other tasks that need to capture the relation between two pieces of text such as Information Retrieval, Question Answering, etc. Furthermore, for the task of sentence ordering, our conditional representation is based on previous selected sentences which means that it can capture the relation in one direction. It would be interesting to extend this approach to condition the sentence representation on both previous and following sentences in the text to achieve a richer representation. We believe that using such a representation can further improve the performance of sentence ordering models.

We also showed that pre-trained Language Models (LMs) can address the task of surface realization. Our approach achieves state-of-the-art performance in the deep track of the MSR shared tasks (Belz et al., 2020). We also improved the performance by pre-training LMs on synthetic data and then fine-tuning them on the manually labeled data. One of the advantages of this approach is its simplicity where there is no need to design the model for any specific task or language. As future work, it would be interesting to evaluate this approach on languages other than English and even further investigate the performance of multilingual pre-trained language model on all languages to evaluate whether the knowledge learned from one language can be beneficial or transferred to other languages.

Our experiments showed that Language Models are the superior models for the tasks of surface realization in the deep surface realization track. However, their performance are below the state-of-the-art in the shallow track. By manually analysing the errors, we found that many are related to long and rare words (e.g. URLs or IP addresses). One solution would be to enhance the copy mechanism (Gu, Lu, Li, & Li, 2016) to avoid misspelled words. The copy mechanism at each decoding step, would decide whether to generate a token or copy it from the input sequence. Another approach that showed promising results is to control the generation by limiting the possible words in the generation step. In this approach, the model can only generate words that belong to a set of pre-defined tokens. This set can be composed of all the tokens in the input augmented with all the inflected forms of those words. As for future work, one can extend the approach with the suggested post processing approaches to control the freedom of the model to generate unexpected tokens.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015, May). Neural machine translation by jointly learning to align and translate. In *Proceedings of Third International Conference on Learning Representations (ICLR 2015).* San Diego, California.

Barzilay, R., Elhadad, N., & McKeown, K. (2001). Sentence Ordering in Multidocument Summarization. In *Proceedings of the First International Conference on Human Language Technology Research (HLT-2001).* USA, San Diego: Association for Computational Linguistics.

Barzilay, R., & Lapata, M. (2008). Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, *34*(1), 1–34.

Barzilay, R., & Lee, L. (2004, May 2 - May 7). Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: (HLT-NAACL 2004)* (pp. 113–120). Boston, Massachusetts, USA.

Basu Roy Chowdhury, S., Brahman, F., & Chaturvedi, S. (2021, November). Is everything in order? a simple way to order sentences. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-2021)* (pp. 10769–10779). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Bechhofer, S. (2009). Owl: Web ontology language. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of Database Systems* (pp. 2008–2009). Boston, MA: Springer US.

Belz, A., Bohnet, B., Ferreira, T. C., Graham, Y., Mille, S., & Wanner, L. (2020, December). Proceedings of the Third Workshop on Multilingual Surface Realisation. Barcelona, Spain (Online): Association for Computational Linguistics.

Bollegala, D., Okazaki, N., & Ishizuka, M. (2005). A Machine Learning Approach to Sentence Ordering for Multidocument Summarization and Its Evaluation. In *Second International Joint Conference on Natural Language Processing (IJCNLP-2005).*

Burstein, J., Tetreault, J., & Andreyev, S. (2010, June). Using entity-based features to model coherence in student essays. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2010)* (pp. 681–684). Los Angeles, California: Association for Computational Linguistics.

Cabezudo, M. A. S., & Pardo, T. (2020, December). NILC at SR'20: Exploring pre-trained models in surface realisation. In *Proceedings of the Third Workshop on Multilingual Surface Realisation (MSR-2020)* (pp. 50–56). Barcelona, Spain (Online): Association for Computational Linguistics.

Chen, X., Qiu, X., & Huang, X. (2016). Neural sentence ordering. *CoRR*, *abs/1607.06952*.

Cianflone, A., & Kosseim, L. (2018, May). Attention for implicit discourse relation recognition. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan: European Language Resources Association (ELRA).

Cui, B., Li, Y., Chen, M., & Zhang, Z. (2018, October-November). Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)* (pp. 4340–4349). Brussels, Belgium.

Cui, B., Li, Y., & Zhang, Z. (2020, November). BERT-enhanced relational sentence ordering network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP-2020)* (pp. 6310–6320). Online: Association for Computational Linguistics.

de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., & Manning, C. D. (2014, May). Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)* (pp. 4585–4592). Reykjavik, Iceland: European Languages Resources Association (ELRA).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT-2019)* (pp. 4171–4186). Minneapolis, Minnesota: Association for

Computational Linguistics.

Du, W., & Black, A. W. (2019, nov). Learning to order graph elements with application to multilingual surface realization. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)* (pp. 18–24). Hong Kong, China: Association for Computational Linguistics.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `https://archive.ics.uci.edu/`

Duan, N., Tang, D., Chen, P., & Zhou, M. (2017). Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP-2017)* (pp. 866–874). Copenhagen, Denmark.

Dušek, O., Novikova, J., & Rieser, V. (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *Computer Speech & Language*, *59*, 123-156.

Elder, H. (2020, December). ADAPT at SR'20: How preprocessing and data augmentation help to improve surface realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation (MSR-2020)* (pp. 30–34). Barcelona, Spain (Online): Association for Computational Linguistics.

Elder, H., & Hokamp, C. (2018, July). Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation (MSR-2018)* (pp. 49–53). Melbourne, Australia: Association for Computational Linguistics.

Fan, A., Gardent, C., Braud, C., & Bordes, A. (2019, November). Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)* (pp. 4186–4196). Hong Kong, China: Association for Computational Linguistics.

Farahnak, F., & Kosseim, L. (2021). Using conditional sentence representation in pointer networks for sentence ordering. In (p. 288-295). California, USA (Online).

Farahnak, F., & Kosseim, L. (2022, December). Pre-training language models for surface realization. In *Proceedings of the 5th International Conference on Natural Language and Speech*

Processing (ICNLSP 2022) (pp. 312–318). Trento, Italy: Association for Computational Linguistics.

Farahnak, F., Mohammadi, E., Davari, M., & Kosseim, L. (2021, 6 8). Semantic similarity matching using contextualized representations. *Proceedings of the Canadian Conference on Artificial Intelligence*.

Farahnak, F., Rafiee, L., Kosseim, L., & Fevens, T. (2019). The Concordia NLG Surface Realizer at SRST 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)* (pp. 63–67). Hong Kong, China.

Farahnak, F., Rafiee, L., Kosseim, L., & Fevens, T. (2020, December). Surface realization using pretrained language models. In *Proceedings of the Third Workshop on Multilingual Surface Realisation (MSR-2020)* (pp. 57–63). Barcelona, Spain (Online): Association for Computational Linguistics.

Glahn, H. R. (1970). Computer-produced worded forecasts. *Bulletin of the American Meteorological Society*, *51*(12), 1126 - 1132.

Gong, J., Chen, X., Qiu, X., & Huang, X. (2016). End-to-end neural sentence ordering using pointer network. *CoRR*, *abs/1611.04953*.

Graham, Y., Baldwin, T., Moffat, A., & Zobel, J. (2017). Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, *23*, 3-30.

Gu, J., Lu, Z., Li, H., & Li, V. O. (2016, August). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-2016)* (pp. 1631–1640). Berlin, Germany: Association for Computational Linguistics.

Harkous, H., Groves, I., & Saffari, A. (2020, December). Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING-2020)* (pp. 2410–2424). Barcelona, Spain (Online): International Committee on Computational Linguistics.

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS-2015)* (p. 1693–1701).

Cambridge, MA, USA: MIT Press.

Hobbs, J. (1990). *Literature and cognition* (No. 21). Center for the Study of Language (CSLI).

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hovy, E., van Noord, G., Neumann, G., & Bateman, J. (1996). Language Generation. *Survey of the State of the Art in Human Language Technology*, 131–146.

Hu, J., Cheng, Y., Gan, Z., Liu, J., Gao, J., & Neubig, G. (2020, Apr.). What makes a good story? designing composite rewards for visual storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 7969-7976.

Huang, T.-H. K., Ferraro, F., Mostafazadeh, N., Misra, I., Devlin, J., Agrawal, A., ... Batra, D. (2016). Visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016).*

Iter, D., Yoon, J., & Jurafsky, D. (2018, June). Automatic Detection of Incoherent Speech for Diagnosing Schizophrenia. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic* (pp. 136–146). New Orleans, LA: Association for Computational Linguistics.

Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing (3nd Edition)*. USA. Retrieved 2022-09-01, from `https://web.stanford.edu/~jurafsky/slp3/`

Kale, M., & Rastogi, A. (2020, December). Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation (INLG-2020)* (pp. 97–102). Dublin, Ireland: Association for Computational Linguistics.

Konstas, I., & Lapata, M. (2012, July). Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)* (pp. 369–378). Jeju Island, Korea.

Kovács, Á., Ács, E., Ács, J., Kornai, A., & Recski, G. (2019, November). BME-UW at SRST-2019: Surface realization with interpreted regular tree grammars. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)* (pp. 35–40). Hong Kong, China: Association for Computational Linguistics.

Kumar, P., Brahma, D., Karnick, H., & Rai, P. (2020, April). Deep attentive ranking networks for learning to order sentences. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, *34*(05), 8115-8122.

Lapata, M. (2003, July). Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)* (pp. 545–552). Sapporo, Japan: Association for Computational Linguistics.

Läubli, S., Sennrich, R., & Volk, M. (2018, October-November). Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP-2018)* (pp. 4791–4796). Brussels, Belgium: Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., … Zettlemoyer, L. (2020, July). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL-2020)* (pp. 7871–7880). Online: Association for Computational Linguistics.

Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., & Dolan, B. (2016, August). A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-2016)* (pp. 994–1003). Berlin, Germany: Association for Computational Linguistics.

Li, J., & Hovy, E. (2014, October). A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)* (pp. 2039–2048). Doha, Qatar: Association for Computational Linguistics.

Liu, X., Wang, X., & Matwin, S. (2018). Interpretable deep convolutional neural networks via meta-learning. In *2018 International Joint Conference on Neural Networks (IJCNN 2018)* (p. 1-9).

Logeswaran, L., Lee, H., & Radev, D. R. (2018). Sentence ordering and coherence modeling using recurrent neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)* (pp. 5285–5292). New Orleans, Louisiana, USA.

Loshchilov, I., & Hutter, F. (2019, May). Decoupled weight decay regularization. In *International*

Conference on Learning Representations (ICLR-2019). Ernest N. Morial Convention Center, New Orleans.

Madsack, A., Heininger, J., Davaasambuu, N., Voronik, V., Käufl, M., & Weißgraeber, R. (2018, July). AX semantics' submission to the surface realization shared task 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation (MSR-2018)* (pp. 54–57). Melbourne, Australia: Association for Computational Linguistics.

Mager, M., Fernandez Astudillo, R., Naseem, T., Sultan, M. A., Lee, Y.-S., Florian, R., & Roukos, S. (2020, July). GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL-2020)* (pp. 1846–1852). Online: Association for Computational Linguistics.

Mann, W., & Thompson, S. (1988). Rhetorical structure theory: Towards a functional theory of text organization. *Text*, *8*(3), 243–281.

May, J., & Priyadarshi, J. (2017, August). SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 536–545). Vancouver, Canada: Association for Computational Linguistics.

Mazzei, A., & Basile, V. (2019). The dipinfounito realizer at srst'19: Learning to rank and deep morphology prediction for multilingual surface realization. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)* (pp. 81–87).

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2017, April). Pointer sentinel mixture models. In *5th International Conference on Learning Representations, (ICLR 2017).* Toulon, France: OpenReview.net.

Mille, S., Belz, A., Bohnet, B., Graham, Y., & Wanner, L. (2019, November). Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019). Hong Kong, China: Association for Computational Linguistics.

Mille, S., Belz, A., Bohnet, B., Pitler, E., & Wanner, L. (2018, July). Proceedings of the First Workshop on Multilingual Surface Realisation. Melbourne, Australia: Association for Computational Linguistics.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., ... Allen, J. (2016, June). A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-2016)* (pp. 839–849). San Diego, California: Association for Computational Linguistics.

Nallapati, R., Zhai, F., & Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)* (pp. 3075–3081). San Francisco, California, USA.

Oh, B., Seo, S., Shin, C., Jo, E., & Lee, K.-H. (2019, November). Topic-guided coherence modeling for sentence ordering by preserving global and local information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP-2019)* (pp. 2273–2283). Hong Kong, China: Association for Computational Linguistics.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., ... Auli, M. (2019, June). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT 2019): Demonstrations* (pp. 48–53). Minneapolis, Minnesota: Association for Computational Linguistics.

Pan, J. Z. (2009). Resource description framework. In *Handbook on ontologies* (pp. 71–90). Springer.

Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL-2020)* .

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8).

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1-67.

Recski, G., Kovács, Á., Gémes, K., Ács, J., & Kornai, A. (2020, December). BME-TUW at SR'20: Lexical grammar induction for surface realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation (MSR-2020)* (pp. 21–29). Barcelona, Spain (Online): Association for Computational Linguistics.

Reiter, E., & Dale, R. (2000). *Building Natural Language Generation Systems*. New York, NY, USA: Cambridge University Press.

Singh, S., Sharma, A., Chawla, A., & Singh, A. (2018, July). IIT (BHU) varanasi at MSR-SRST 2018: A language model based approach for natural language generation. In *Proceedings of the First Workshop on Multilingual Surface Realisation (MSR-2018)* (pp. 29–34). Melbourne, Australia: Association for Computational Linguistics.

Somasundaran, S., Burstein, J., & Chodorow, M. (2014, August). Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2014,)* (pp. 950–961). Dublin, Ireland: Dublin City University and Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems (NIPS 2017)* (pp. 5998–6008). Long Beach Convention Center, Long Beach.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations (ICLR 2018).*

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer Networks. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2692–2700.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 3156-3164.

Wang, T., & Wan, X. (2019, 07). Hierarchical attention networks for sentence ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-2019)* (Vol. 33, p. 7184-7191). Honolulu, Hawaii, USA.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, *abs/1910.03771*.

Yang, Y., Yih, W.-t., & Meek, C. (2015, September). WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)* (pp. 2013–2018). Lisbon, Portugal: Association for Computational Linguistics.

Yin, Y., Song, L., Su, J., Zeng, J., Zhou, C., & Luo, J. (2019). Graph-based neural sentence ordering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI-19)* (pp. 5387–5393). Macao, China.

Yu, X., Falenska, A., Haid, M., Vu, N. T., & Kuhn, J. (2019). Imsurreal: Ims at the surface realization shared task 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019) (EMNLP 2019)* (pp. 50–58). Hong Kong, China.

Yu, X., Tannert, S., Vu, N. T., & Kuhn, J. (2020, December). IMSurReal too: IMS in the surface realization shared task 2020. In *Proceedings of the Third Workshop on Multilingual Surface Realisation (MSR-2020)* (pp. 35–41). Barcelona, Spain (Online): Association for Computational Linguistics.

Zeman, D., Ginter, F., Hajič, J., Nivre, J., Popel, M., & Straka, M. (2018). CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies.* Brussels, Belgium: Association for Computational Linguistics.

Zhou, Y., Liu, C., & Pan, Y. (2016, December). Modelling sentence pairs with tree-structured attentive encoder. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING 2016)* (pp. 2912–2922). Osaka, Japan.