

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
Programa de Pós-graduação em Informática

João Ribeiro Bezerra

**NEWRITER: A TEXT EDITOR FOR BOOSTING  
CREATIVE SCIENTIFIC PAPER WRITING**

Belo Horizonte  
2021

João Ribeiro Bezerra

**NEWRITER: A TEXT EDITOR FOR BOOSTING  
CREATIVE SCIENTIFIC PAPER WRITING**

Dissertação apresentada ao Programa de Pós-graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Luís Fabrício Wanderley Góes

Área de concentração: Ciência da Computação

Belo Horizonte  
2021

#### FICHA CATALOGRÁFICA

Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

B574n	<p>Bezerra, João Ribeiro</p> <p>NeWriter: a text editor for boosting creative scientific paper writing / João Ribeiro Bezerra. Belo Horizonte, 2021.</p> <p>38 f. : il.</p> <p>Orientador: Luís Fabrício Wanderley Góes</p> <p>Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática</p> <p>1. Editor de textos (Programas de computador). 2. Processamento de linguagem natural (Computação). 3. Escrita. 4. Texto acadêmico. 5. Software de sistemas. 6. Word (Programa de computador). 7. Comunicação na tecnologia. 8. Redação técnica. I. Góes, Luís Fabrício Wanderley. II. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática. III. Título.</p> <p>SIB PUC MINAS</p> <p>CDU: 681.3.06</p>
-------	--

João Ribeiro Bezerra

**NEWRITER: A TEXT EDITOR FOR BOOSTING  
CREATIVE SCIENTIFIC PAPER WRITING**

Dissertação apresentada ao Programa de Pós-graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática.

Área de concentração: Ciência da Computação

---

Prof. Dr. Luís Fabrício Wanderley Góes(Orientador) – PUC Minas

---

Prof. Dr. Wladimir Cardoso Brandão– PUC Minas

---

Prof. Dr. Luiz Chaimowicz– Universidade Federal de Minas Gerais

Belo Horizonte, 19 de Maio de 2021

*Dedico este trabalho a todos que me ajudaram em seu desenvolvimento. À minha esposa Flávia, à minha mãe Elza, à minha irmã Luísa, ao meu orientador Prof. Dr. Luís Fabrício W. Góes, ao Prof. Dr. Wladimir C. Brandão, e à secretária do Programa, Giovana. Muito obrigado a todos por todo o apoio.*

## **AGRADECIMENTOS**

Agradeço à CAPES, PUC Minas, CNPq, e FAPEMIG pelo suporte financeiro para a realização deste trabalho.

*“As long as you have the desire to do so, you can learn from anyone or any book, no matter what the topic. Lessons are all around us, waiting to be learned.” (UMEHARA, 2016)*

## RESUMO

Atualmente, no campo científico, a produção de textos é necessária para que cientistas compartilhem suas pesquisas e contribuições para a ciência. A escrita de textos científicos é uma tarefa que demanda tempo e habilidades de escrita formal e pode ser lenta e desafiadora, especialmente para pesquisadores inexperientes. Além disso, os textos científicos devem ser escritos em inglês, seguir um estilo específico e utilizar termos específicos, o que pode ser uma tarefa difícil especialmente para pesquisadores que não são falantes nativos do inglês ou que não conhecem os procedimentos específicos de redação exigidos por alguma editora. No entanto, um simples assistente de escrita pode induzir à escrita de textos muito semelhantes ou até mesmo induzir ao plágio, o que motivou o uso de Criatividade Computacional. Neste estudo, é proposta uma ferramenta para abordar a assistência à escrita criativa de textos científicos. Ele permite que os usuários utilizem texto científico relacionado como entrada para treinar um modelo de linguagem base, treinado em texto científico, em um modelo especializado e personalizado pelo usuário. Em seguida, sugestões de texto criativas em tempo real são exibidas ao usuário à medida que ele escreve seu próprio texto, relacionadas ao contexto do texto e criativas. Neste estudo, propomos o NeWriter, um *framework* para sugestões textuais criativas e personalizadas pelo usuário. Os resultados deste estudo mostram que modelos de linguagem personalizados pelo usuário podem ser usados para melhorar sua eficácia na assistência à escrita de textos científicos em comparação com modelos pré-treinados do estado da arte, e detalha sugestões textuais criativas utilizando modelos de linguagem do estado da arte.

Palavras-chave: Assistência à Escrita. Modelos de Linguagem. Processamento de Linguagem Natural. Criatividade Computacional.



## ABSTRACT

Nowadays, in the scientific field, text production is required from scientists as means of sharing their research and contribute to science. Scientific text writing is a task that demands time and formal writing skills and can be specifically slow and challenging for inexperienced researchers. Moreover, scientific texts must be written in English, follow a specific style, and use specific terms, which can be a difficult task specially for researchers that aren't native English speakers or that don't know the specific writing procedures required by some publisher. However, a naive writing assistant can also result in similar texts or even plagiarism, which motivated the use of Computational Creativity. In this study, a tool is proposed for addressing creative scientific text writing assistance. It enables users to feed related scientific text as an input to train a scientific-text trained base Language Model into a user-customized, specialized one. Then, the user is presented with creative real-time text suggestions as they write their own text, which are related to the text's context while also being creative. In this study we propose NeWriter, a framework for user-customized, creative textual suggestions. This study's results show that user-customized language models can be used to improve their effectiveness for scientific text writing assistance compared to state-of-the-art pre-trained models, and details creative textual suggestions using state-of-the-art Language Models.

Keywords: Writing Assistance. Language Models. Natural Language Processing. Computational Creativity.

## LIST OF FIGURES

FIGURE 1 –	Vector offsets in word embeddings. . . . .	18
FIGURE 2 –	Seq2seq architecture. . . . .	19
FIGURE 3 –	The Transformer architecture. . . . .	20
FIGURE 4 –	Transformer’s encoder and decoder stacks. . . . .	21
FIGURE 5 –	NEWRITER’s architecture. . . . .	25
FIGURE 6 –	Newriter’s architecture with added Computational Creativity Module. . . . .	26
FIGURE 7 –	Writing software example 1. . . . .	27
FIGURE 8 –	Writing software example 2. . . . .	28
FIGURE 9 –	Example of creative textual recommendations. . . . .	32
FIGURE 10 –	Example of creative textual recommendations in an abstract. . . . .	32
FIGURE 11 –	Example of creative textual recommendations in an abstract. . . . .	33
FIGURE 12 –	Average perplexity of the textual recommendations for base NeWriter, a language model trained using classical English literature books (Books), and SciBERT for each word in a scientific text’s abstract. . . . .	33
FIGURE 13 –	Number of new words the language model trained with classical English literature books suggested comparing its 10 best rated suggestions to base NeWriter and SciBERT, for each word in a scientific text’s abstract. . . . .	34

## LIST OF TABLES

TABLE 1	–	Perplexity comparison between the User-customized LM and Sci-BERT for specific domain scientific text . . . . .	31
---------	---	---	----

## LIST OF CHARTS

## LIST OF ABBREVIATIONS

LM – Language Model  
NLP – Natural Language Processing  
seq2seq – Sequence-to-Sequence  
RDC – Regent-Dependent Creativity

## SUMMARY

<b>1 INTRODUCTION</b>	<b>14</b>
1.1 Motivation	15
1.2 Objective	15
1.2.1 <i>Specific objectives</i>	15
1.3 Justificative	16
1.4 Contributions	16
1.5 Document organization	16
<b>2 BACKGROUND</b>	<b>17</b>
2.1 Computational Creativity	17
2.2 Word embedding	18
2.3 Sequence-to-sequence (seq2seq) models	18
2.4 The Transformer architecture	19
2.5 BERT - Bidirectional Encoder Representations from Transformers	21
2.6 SciBERT Language Model	22
<b>3 RELATED WORK</b>	<b>23</b>
<b>4 NEWRITER</b>	<b>25</b>
4.1 Computational Creativity Module	26
<b>5 METHODOLOGY</b>	<b>29</b>
5.1 Tools	29
5.2 Methods and Metrics	29
5.2.1 <i>Perplexity</i>	29
5.2.2 <i>User customized language model evaluation</i>	29
5.3 Computational Creativity Module evaluation	30
5.4 Value	30
5.5 Diversity	30
5.6 Creativity (RDC)	30
<b>6 EXPERIMENTAL RESULTS</b>	<b>31</b>
6.1 User-customized Language Model evaluation	31
6.2 Computational Creativity	31
<b>7 CONCLUSION</b>	<b>35</b>
<b>REFERENCES</b>	<b>37</b>

## 1 INTRODUCTION

Text production is required from scientists as a mean of sharing their research and contribute to science. Scientific text writing is a task that demands time and formal writing skills. The writing process can be specifically slow and challenging for inexperienced researchers, delaying the release of their research results (ITO et al., 2019). Additionally, most scientific texts must be written in English, following a specific style, and using specific terminology to be accepted for publishing in respectable journals and conferences. This can be a difficult task specially for non native speakers or those that don't know the specific writing procedures required by some publishers, demanding extra effort and time. Therefore, for a scientist in formation it is specially hard to begin producing scientific text until they earn more experience and skills.

Considering the importance of text production, the Natural Language Processing (NLP) area is widely studied by the scientific community in many applications to solve some of natural language problems. Recent advances in NLP come as result of the proposal of Sequence-to-Sequence (seq2seq) language models and the Transformer architecture (VASWANI et al., 2017). These model and neural network architecture address the text processing as a sequence of words, keeping the context of other words in the text when processing each singular word. The sequences of words are transformed into embeddings that are processed by the Language Model (LM) by using Deep Learning (DL) algorithms. Therefore, many approaches have been proposed for problems such as machine translation (LI; JIANG; LIU, 2019), text summarization (KIEUVONGNGAM; TAN; NIU, 2020), speech recognition (LIU et al., 2020), ancient text restoration (ASSAEL; SOMMERSCHIELD; PRAG, 2019), and question-answering (ESTEVA et al., 2020). These approaches have been successful in these tasks, even being used in a production environment by huge companies, such as Google (CHEN et al., 2019). More recently, more advanced Language Models such as the GPT-3 (BROWN et al., 2020) have been developed and are showing great improvements over the most used Language Models, which shows interest and development in the area.

A lot of effort is being done into developing tools for writing assistance. Previous studies show positive results for this application, such as text auto-completion based on the user's writing patterns and style (CHEN et al., 2019) and scientific text writing assistance for non-native English researchers (ITO et al., 2019). Furthermore, with the proposal of LMs compliant to parallel computing, NLP solutions are becoming faster and more reliable. Along with the faster processing times comes the possibility of development of real-time solutions for natural language problems, such as text writing assistance (DONAHUE; LEE; LIANG, 2020).

Another topic that has been the focus of many recent studies is Computational Creativity. The main goal of Computational Creativity is to generate artefacts that would be considered creative if they were created by a human, which has great impact over human perception in computer-generated content. Computational Creativity is used for the generation of artistic artefacts, such as painting and music generation (BODEN, 2009); generation of culinary recipes (SANTOS et al., 2020); storytelling in electronic games (AMMANABROLU et al., 2020); and in the generation of poems (CRUYS, 2020).

Section 1.1 details this work's motivation, Section 1.2 details the objective, Section 1.3 contains the justificative, Section 1.4 contains this work's contributions, and Section 1.5 details this document's organization.

## 1.1 Motivation

Although a lot of work has been done in the context of writing assistance (CHEN et al., 2019; DONAHUE; LEE; LIANG, 2020), not many of them are focused on scientific writing assistance, such as (ITO et al., 2019). Additionally, other studies in this area don't specifically focus on real-time text suggestions. Moreover, these studies don't account for user customization when it comes to sub-area specific terms and writing style. In the subject of scientific writing assistance, the user should be assisted for faster, higher quality text writing, while also having compliance with the sub-area's specific terms and writing style.

The usage of a writing assistance tool such as NeWriter helps users write a text that follows the writing style of the input text. However, as the perplexity of NeWriter's language model gets lower, the texts generated by users tend to get too similar to the input text. This can lead NeWriter to encourage plagiarism and to create more repetitive scientific texts. In order to avoid repetitive texts, we propose the use of Computational Creativity. With the use of Computational Creativity, NeWriter can display to the user not only text with the best score, but also recommendations that, while still having a good score, encourage more diversity in the user's text. To do this, NeWriter can generate an initial textual recommendation using a Language Model trained with scientific texts, and then swap the recommendation's words using other Language Models, trained with texts from other domains, to increase its vocabulary range and diversity. However, not all word swaps will be proper for the scientific text context, so we can use the original Language Model as a proxy to measure the quality of the creative suggestions made by the creative Language Model.

## 1.2 Objective

The objective of this work is to assist in the creative writing of research papers for specific niches. This is two-fold: i) current ML-based assistants focus on accuracy rather than diversity, which might lead to plagiarism or non-creative text writing, impacting text writing evolution of a certain field; ii) and they also do not take into account that each conference, journal or field have some peculiarities (jargons, syntactic structure, terminologies etc.) that makes a paper more suitable (familiar) for that venue.

To achieve this goal, we proposed a tool that, given a number of input scientific texts related to a certain area of interest, it generates recommendations to assist the user in the process of writing scientific text. The text suggestions must be compliant with specific terms used in the sub-area the user is writing for, enabled by user customization. Furthermore, the resulting text should be diversified to avoid direct copies of parts of other texts. One additional positive outcome for these creative suggestions is that they can help with writer's block, as they show different words the user can pick from without losing their creative flow.

### 1.2.1 *Specific objectives*

Therefore, for this study we propose a tool based on Huggingface's Transformers language model for real-time scientific text writing assistance. In order to reach this objective and evaluate the proposed approach, the following specific objectives are posed:

- Propose a learning approach that the user can fine-tune the language model training by using other scientific texts as input for customized writing assistance;



- Propose a real-time writing assistant, where the approach shows text suggestions as the user writes their text.
- Assert the quality and diversity of the suggestions made by the Language Model.

### 1.3 Justificative

In the NLP area, writing assistance is a topic focused by many studies. The state-of-the-art techniques such as seq2seq (Sequence-to-Sequence) models and the Transformer architecture are widely used since their proposal, and can help solve many challenges in the area (ITO et al., 2019) (WANG; CHO, 2019).

The Transformer architecture fits with the problem addressed in this research because it allows for the understanding of language patterns. And the addressed problem's domain lies on scientific texts, which follow a structured pattern, built with normalized writing patterns and formal terms as a standard. Therefore, the use of NLP, specially with tools such as Transformer language models, shows great potential in addressing the challenges this study proposes to investigate.

In addition, the proposed approach differs from previous works by the fact that it allows for user customization through input scientific texts, which results in more accurate assistance in terms of the user's area of study or in the specific writing style used in a publication medium. The proposed approach also aims to explore the negative effects of text writing assistance, making use of computational creativity to avoid textual plagiarism and assist in more diversified text writing.

### 1.4 Contributions

In this work, we developed a user-customizable tool that generates creative suggestions to help scientific text writing. The user is able to feed the system with reference work (i.e. related papers, papers from the same conference) to customize NeWriter's Language Model. The customized Language Model is used in conjunction to a creative Language Model, trained using literary texts, to add diversity to NeWriter's textual recommendations.

A paper with partial results of this work was published in the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) as a paper entitled NEWITER: A Text Editor for Boosting Scientific Paper Writing.

### 1.5 Document organization

This document is organized as follows. Chapter 2 presents the theoretical foundation. Chapter 3 presents the related work. Chapter 4 presents the proposed tool, NeWriter. Chapter 5 presents the methodology, with the used tools, methods and metrics. Chapter 6 contains the experimental results. Chapter 7 contains the conclusion.

## 2 BACKGROUND

The development of NLP techniques is essential for the current advances in its applications, such as Language Understanding, Language Modelling, Machine Translation, etc. It provides researchers with tools capable of better representations of language models that can learn language better, process input faster, and reach more accurate results.

This chapter is organized as follows: Computational Creativity is discussed in Section 2.1, Word embeddings are described in Section 2.2, Sequence-to-sequence models are presented in Section 2.3, the Transformer architecture is shown in Section 2.4, the BERT Language Model is described in Section 2.5.

### 2.1 Computational Creativity

Computational Creativity is the area of study in Artificial Intelligence that aims to use computational systems to generate artifacts that would be considered creative if they were human-created. There are two approaches to achieve Computational Creativity: one is focused on recreate the human process of creation in order to achieve creativity; the other approach focuses on the generated artefact, in which one tries to generate results that would be perceived as creative by humans. This work focuses on the generated artefact (BODEN, 2009) (SANTOS et al., 2020).

Computational Creativity can be used both for artistic artefacts, such as painting, music generation; and scientific theories, mathematical concepts, and scientific projects (BODEN, 2009). Computational Creativity also has been used for the generation of culinary recipes (SANTOS et al., 2020), storytelling in electronic games (AMMANABROLU et al., 2020), and in the generation of poems (CRUYS, 2020).

A creative artefact is a new and valuable artefact. The area explores concepts such as novelty, surprise and value for measuring generated artefacts. An artefact's novelty is related to the difference between its characteristics and the existing characteristics in a group of known artefacts. However, the known artefacts are context-dependant, as different individuals may know only a subgroup of the existing artefacts. This fact leads into two definitions of creativity: P-creativity (Psychological, when novelty is determined by a group of individuals' known artefacts), and H-creativity (Historical, when novelty is determined by all the individuals' known artefacts). Therefore, H-creativity is a special case of P-creativity (BODEN, 2009).

A common metric used to calculate surprise is using the Bayesian surprise metric, based on the difference in the level of information before and after the observation of an event. In generated artefacts, this means the difference between the previously known universe of possible artefacts in comparison to the knowledge after the generation (BALDI; ITTI, 2010).

Another way of measuring the creativity of artefacts is using the domain-independent metric RDC (Regent-Dependent Creativity) metric. Proposed by (FRANÇA et al., 2016), it is calculated as shown in Equation 2.1, where for an artefact  $a$ , its creativity is the sum of the normalized novelty  $n_a$  and value  $v_a$  ( $s_a$ ) plus an extra penalty term. The penalization is needed to avoid considering creative the artefacts with high novelty and low value (different but useless artefacts) or the artefacts with high value and low novelty (most are already known artefacts). Equation 2.2 is used for the penalty term, which penalized an artefact depending on the difference among its novelty and value ( $d_a$ ). The penalty is proportional to the difference between novelty and value. The creativity is in the range  $[0,2]$  (FRANÇA et al., 2016).

$$rdc(a) = n_a + v_a - p(n_a, v_a) \quad (2.1)$$

$$p(n_a, v_a) = s_a(1 - e^{-kd_a}) \quad (2.2)$$

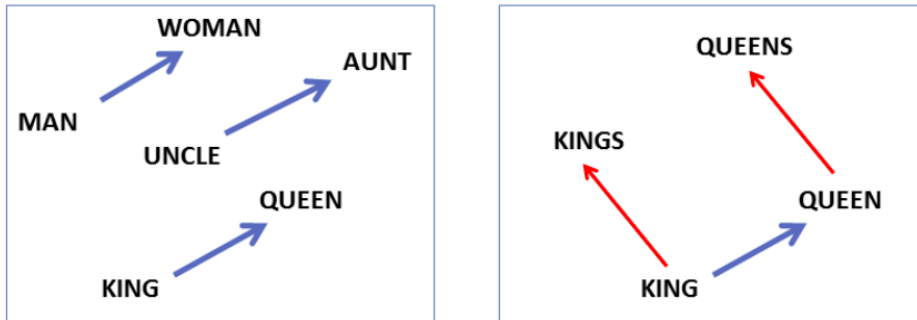
In its design, the RDC metric can be adapted to be used to measure creativity in any given context that can be adapted to fit the equation's format.

## 2.2 Word embedding

NLP is related to other fields such as Artificial Intelligence, and broadly Machine Learning. Recently, state-of-the-art NLP techniques rely heavily on Neural Networks, and that requires numerical vector input. Thus, language models use word embeddings, representations of words in a vector space. These representations project the relations between words in the form of their vector offsets. Figure 1 shows gender relationship between words on the left panel, and singular/plural relationship between words on the right panel (MIKOLOV; YIH; ZWEIG, 2013).

Word embeddings are the base concept used in all Language Models, as they create a framework to represent and relate words to each other, which is of extreme importance when analysing context and meaning of textual artifacts (MIKOLOV; YIH; ZWEIG, 2013).

Figure 1 – Vector offsets in word embeddings



Source: MIKOLOV, YIH E ZWEIG, 2013

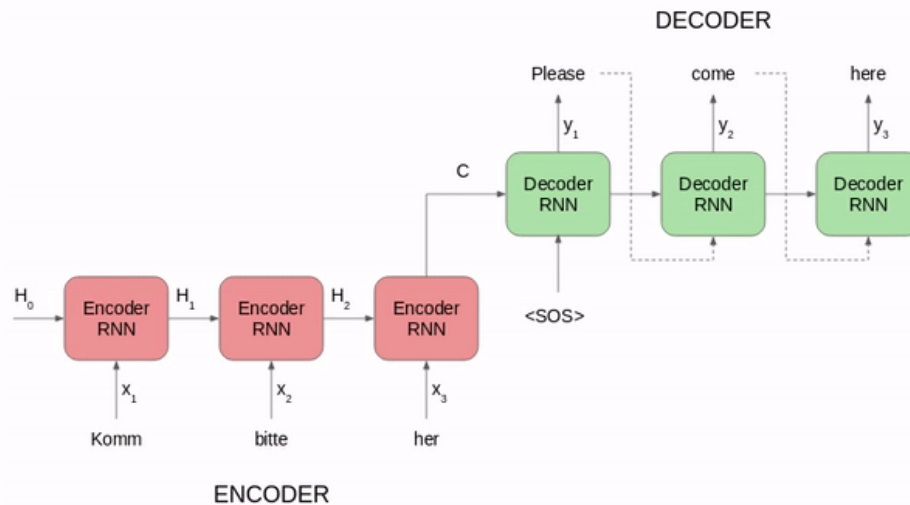
## 2.3 Sequence-to-sequence (seq2seq) models

Sequence-to-sequence (seq2seq) are language models that use recurrent neural network (RNN) to convert an input sequence into an output sequence, and are widely used in NLP since text can be represented as a sequence of words. A common example of this use is machine translation, where a sequence of words in a language is transformed into a sequence of words in another language.

Figure 2 shows that in seq2seq models' architecture, multiple RNNs are created for each word in the input text (both for encoders and decoders). In Figure 2,  $x_i$  are the elements of the word vector and  $H_i$  are the hidden states that work as a context vector. Words are processed in order, and the sum of  $H_i$  is passed from each RNN to the next, in a process known as the attention mechanism. So the context from previous words is used in the processing of the current word in the vector.

In other words, the encoders are RNNs trained to translate the input text into data that contains more dimensions, one example being the other words in the sentence and

Figure 2 – Seq2seq architecture



Source: VASWANI ET AL., 2017

their weights related to that single word. The decoders are RNNs trained to transform this data back to a word sentence, considering all the extra data beyond the input word sequence. In this case, the attention mechanism consists of using the other words in a sentence as context for processing each word, which is used to obtain better results than a naive word by word translation. In some cases, for example, when a word in a language can't be directly translated into a word in other language, the Language Model should be able to output a whole expression that conveys the intended meaning, which is only possible with the use of the extra data computed in the process of encoding and decoding the text sequence.

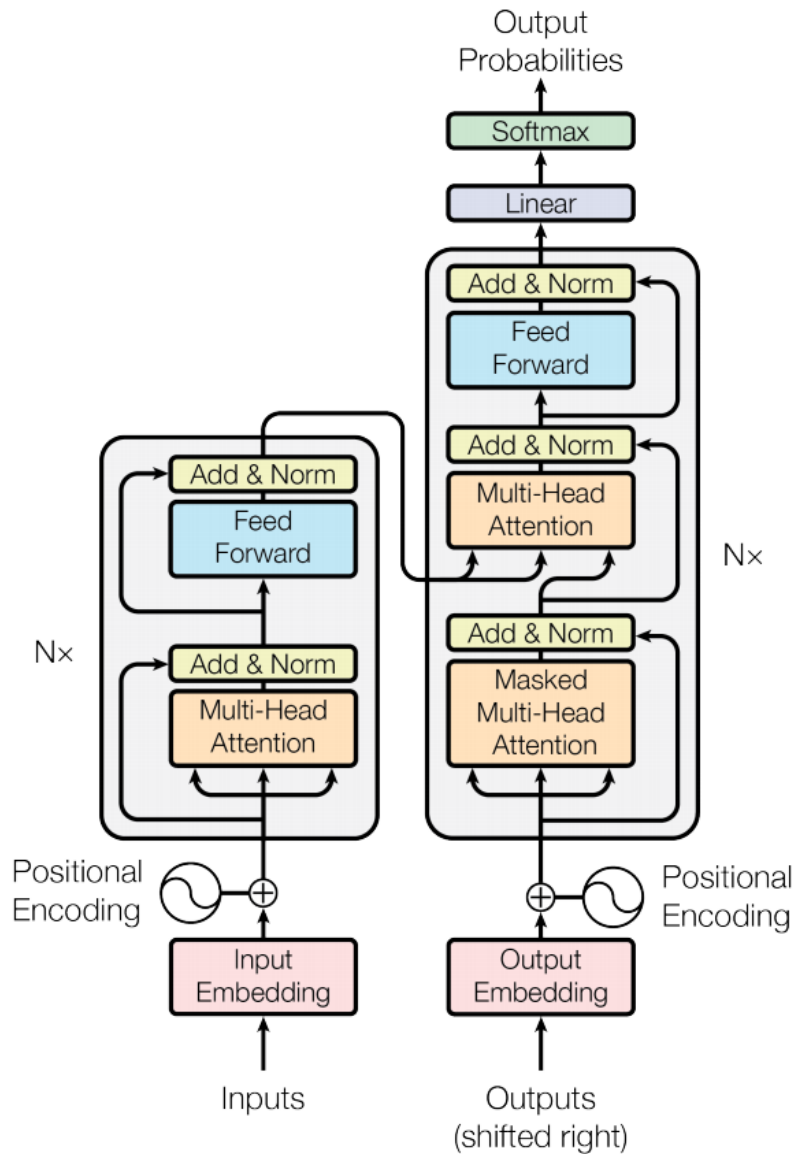
The development of seq2seq models was a breakthrough for textual translation tasks, being used to transform sentences in one language (represented as sequences of words) into sentences in another language.

Although very robust, the sequential processing nature of this architecture prevents parallel processing, which limits its use for longer sequences of text. Another challenge for the use of seq2seq models is the difficulty for the RNNs to learn from the dependencies originated by long-ranged sequences of words on the input vector (VASWANI et al., 2017).

## 2.4 The Transformer architecture

The Transformer architecture was proposed to solve the challenges faced by seq2seq architectures, such as dealing with long-range dependencies. This architecture can handle dependencies between input and output using only recurrence and the attention mechanism. Figure 3 presents the Transformer architecture, with the encoder on its left half and the decoder on its right half.

From Figure 3 we observe that both the encoder and the decoder are composed of  $N = 6$ , respectively identical layers. Each of the encoder's layers has two sub-layers: a multi-head self-attention mechanism, and a position-wise fully-connected feed-forward network. Each of the decoder's layers has, in addition to the two layers in each encoder layer, a multi-head attention module over the output of the encoder stack. The multi-head attention modules are formed by a stack of self-attention modules, an attention mechanism that relates different positions of a sentence in order to compute a representation of

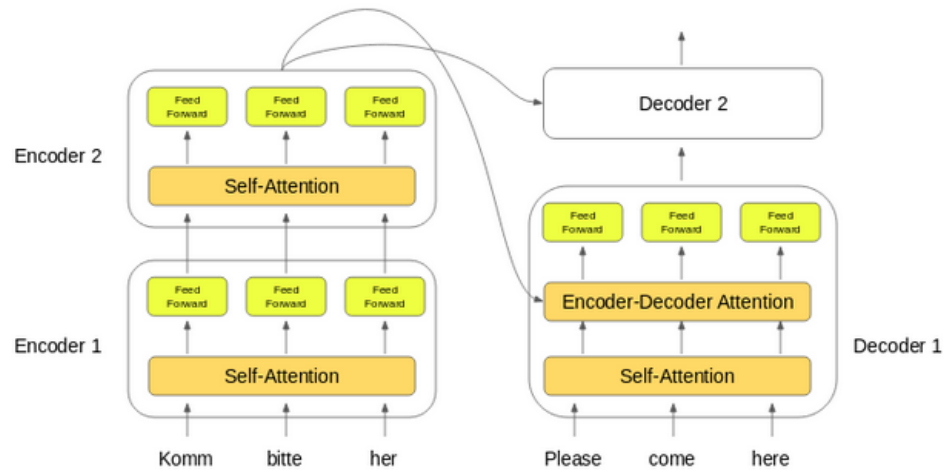
**Figure 3 – The Transformer architecture**

Source: VASWANI ET AL., 2017

the sequence. Multi-head attention modules consist of multiple instances of self-attention modules. The self-attention modules enable the architecture to consider other words of the input sequence to better understand each individual word in the sequence. The self-attention module estimates the importance of each word in the sequence to the current word. These values are estimated multiple times in the Transformer architecture in parallel and independently, hence the name multi-head attention. Figure 4 shows the data flow between encoders and decoders in the Transformer architecture.

From Figure 4 we observe that first, the word embeddings from the input sequence are passed into the first encoder block. The embeddings are transformed and passed on to the next encoder, and this process repeats until the last encoder is reached. The last encoder outputs the result to all the decoders in the stack. However, attention modules can only work with fixed-length strings, meaning the input text must be subdivided before used as input. This causes context fragmentation, which limits the Transformer

Figure 4 – Transformer’s encoder and decoder stacks



Source: VASWANI ET AL., 2017

architecture effectiveness. To overcome this challenge, the Transformer-XL architecture was proposed (DAI et al., 2019). In the Transformer-XL architecture, the hidden states obtained from the previous input fragment from the original text are used in the processing of the next fragment, thereby causing no context fragmentation.

## 2.5 BERT - Bidirectional Encoder Representations from Transformers

The bidirectional encoder representations from Transformers (BERT) is a state-of-the-art language model proposed by Google AI team that uses pre-training and fine-tuning for several NLP tasks. It uses a multi-layer, bidirectional transfer encoder, in which the self-attention layer works on the input sequence in both directions.

The BERT model is pre-trained using two different strategies: masked language modeling and next sentence prediction. In masked language modeling, 15% of an original text words are replaced by either a token "[MASK]" or a random word and are fed for the language model as input. The objective is that these masked tokens should be predicted by the language model. For the next sentence prediction, pairs of sentences are fed for the language model as input, where in 50% of pairs the second sentence is the subsequent sequence in the original text, whereas in the other 50% the second sentence is another random sentence in the text. The objective is to learn and predict if the second sentence fits as subsequent to the first sentence in the original text.

In order to gather huge language understanding, BERT is trained on huge datasets, a process that demands a lot of time and processing. However, once a language model is pre-trained, it can be used for further training in the fine-tuning process. Fine-tuning enables training a language model for specific tasks, while also being able to skip the initial training with the use of knowledge transfer (DEVLIN et al., 2019).

BERT is usually used as a parameter initialiser for other traditional Language Models. In (WANG; CHO, 2019) the authors show that BERT is a Markov random field language model, which enables it to be used for next word prediction in a sentence. The authors show that BERT can be used to generate high-quality text, with slightly worse quality compared to traditional left-to-right models but also with more diversity.

## 2.6 SciBERT Language Model

In order to address the lack of high-quality, large-scale labeled scientific language models, SciBERT was proposed by (BELTAGY; LO; COHAN, 2019). SciBERT is a BERT-based model trained on scientific text on a large multi-domain corpus of scientific publications, aimed to improve the usage of NLP in the scientific text domain. SciBERT is trained on 1.14M papers from Semantic Scholar\* and contains its own vocabulary (scivocab) with 3.1B tokens. Compared to BERT, SciBERT shows better results on scientific domain NLP tasks.

---

\* <http://www.semanticscholar.org>

### 3 RELATED WORK

Several works were reported in literature for writing assistance using NLP. Most of them are related to scientific text writing. The approach proposed by (ITO et al., 2019) is the most similar to NEWWRITER proposal and domain, focusing on scientific writing assistance. However, SmartCompose (CHEN et al., 2019), that aims to assist writing of emails, proposes a learning mechanism very similar to NEWWRITER. Additionally, several studies have been made on the topic of creative text generation, with (CRUYS, 2020) making use of a language model trained with texts from other areas other than their main focus, poetry, to generate poems with more diversity. This strategy is similar to the one proposed with NEWWRITER’s computational creativity module, where text from other domains is used to train a secondary language model to provide more diversity to the textual recommendations.

In (ITO et al., 2019) the authors propose sentence-level revision (SentRev), a writing assistant focused on surface-level issues such as typographical, spelling and grammatical errors. Their system focused on helping inexperienced authors by producing fluent, complete sentences given their incomplete, rough text drafts. The authors also released an evaluation dataset containing incomplete sentences authored by non-native writers along with their final versions in published academic papers that can be used for further research in this area.

In (CHEN et al., 2019) the authors go over details on the implementation of Gmail’s SmartCompose functionality, that generates interactive, real-time suggestions to assist users in writing mails. For the language model selection, they compared state-of-the-art models such as the Transformer architecture and LSTM-based seq2seq models for efficiency and latency. Upon testing, they chose a LSTM-based seq2seq model, since even though the Transformer architecture had more accurate results, its extra computing time wasn’t ideal for their application as the model had more latency as it became more complex. SmartCompose showed high-quality suggestion prediction, enough to be adopted to Google’s platform in production.

In a similar vein, (DONAHUE; LEE; LIANG, 2020) present an approach for text infilling for prediction of missing spans of text at any position in a document. Although the authors cite the potential of masked language infilling for writing assistance tools, their research focuses on language modeling, in the form of infilling text at the end of a document. In this work, the authors fine-tune language models capable of infilling entire sequences on short stories, scientific abstracts and lyrics. A survey showed that humans had difficulty identifying which sentences were machine-generated or original sentences in the short stories domain.

In (AYE; KAISER, 2020), the authors propose a novel design for predicting tokens in real time for source code auto-completion, combining static analysis for in-scope identifiers with the use of a language model, in a system that produces valid code with type-checking. The developed solution achieves state-of-art accuracy while also fitting the constraints of real-world completion implementations in modern IDEs.

In (SHIH; CHANG; YANG, 2019), the authors propose XL-Editor, a training framework for state-of-the-art generalized auto-regressive pre-training methods to revise a given sentence using variable-length insertion probability in order to reflect the nature of how a human revisits a sentence to obtain a refined result. The XL-Editor is able to estimate the probability of inserting a sequence into a specific position of a sentence, execute post-editing operations and complement existing sequence-to-sequence models to refine generated sequences. The authors demonstrated improved post-editing capabilities from



XLNet to XL-Editor. Additionally, XL-Editor is extended to address post-editing style transfer tasks and achieves significant improvement in accuracy, while also maintaining coherent semantic.

In (GRANGIER; AULI, 2018), the authors propose a framework for computer-assisted text editing, applied to translation post-editing and to paraphrasing. In the proposed framework, a human editor modifies a sentence, marking tokens they would like the system to change. Their model then generates a new sentence that reformulates the original sentence by avoiding the marked words. The model was developed using sequence-to-sequence modeling along with a neural network which takes a sentence with change markers as input. Their model is shown to be advantageous for translation post-editing through simulated post-edits and also for paraphrasing through a user study.

The work (SANTOS et al., 2020) explores the generation of culinary recipes with the use of language models, along with their preparation steps. A survey showed that recipes generated by their system had their quality evaluated in a survey as an average of 63.6%, and had one of the recipes cooked with an average taste evaluation of 93%. In this work, the Regent-Dependent Creativity (RDC) metric is used to measure the overall creativity of their model, and the quality of the generations is measured using the perplexity metric, both of which are also used to evaluate NeWriter.

In (AMMANABROLU et al., 2020), in the context of text-adventure games, the authors propose a system to generate semantically coherent quests for the player to tackle. These quests are defined as a series of actions required to progress towards a goal. The generated quests were created using a culinary context, and were evaluated via a survey in comparison with human designed quests. The generated quests showed to be as coherent and implied creativity in a similar fashion to the human designed ones, without loss in perceived quality.

In the study of (CRUYS, 2020), the authors explore the use of language models to generate creative poems. Their system is trained using standard, non-poetic text, while using constraints in order to generate proper poems, such as having verse structures and rhymes. Their system generates poems for both English and French with a quality similar to state of the art models even through the use of a language model not specifically trained to their problem’s domain.

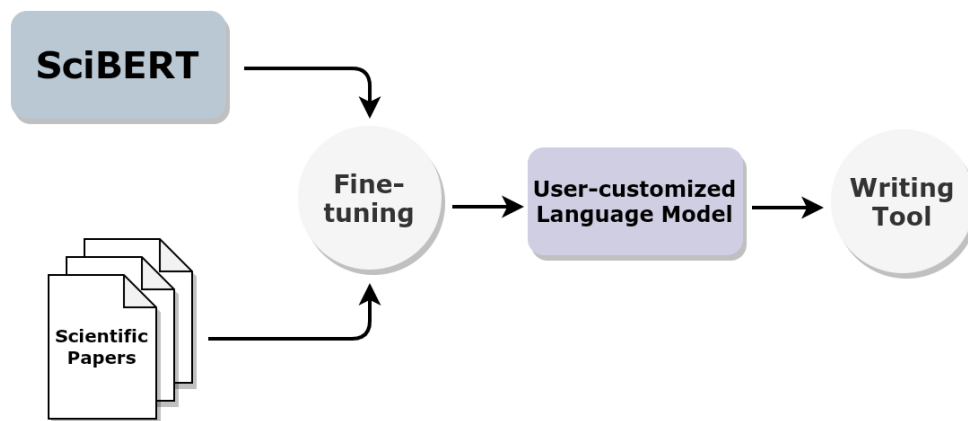
In this section we observe that a lot of progress is being made in the field of NLP with the use of state-of-the-art language models. These works show that modern language models are fit for dealing with tasks related to text writing, writing style learning and reproduction, and text correction. Related work show that language models are able to work with scientific text (ITO et al., 2019) and can be used for real-time writing assistance applications (CHEN et al., 2019). Therefore, the related work also show that the current existing approaches can be used in order to address the problem on scientific paper writing. However, none of the related works implement scientific text language modeling to a real-time writing assistant application, such as NEWWRITER. Additionally, one can also observe that text generation using language models is a subject of several studies, and has been reaching positive results. The related work in the computational creativity domain, most particularly the approach explored in (CRUYS, 2020), is used as inspiration for NEWWRITER’s computational creativity module. However, current ML-based assistants are not focused on diversity, which may lead to plagiarism or non-creative text writing. Current studies also do not take the peculiarities of individual publication medium into account when they generate textual recommendations.

## 4 NEWRITER

Upon research in the literature, the assistance in scientific text writing is a widely studied problem. However, the state-of-the-art NLP tools can be used to address this problem. Moreover, the availability of free usage of computing power to accelerate Machine Learning applications leads to fast and accessible language model training for user-customized solutions.

In this study, we propose NEWRITER, a neural network based approach to address creative scientific text writing assistance. Figure 5 presents the NEWRITER architecture.

**Figure 5 – NEWRITER’s architecture**



**Source: Author**

As shown in Figure 5, initially, the user gathers a number of scientific articles from their area of study. These articles are then used in the process of fine-tuning a base language model in order to customize it to the user’s needs. The language model used as a base was SciBERT, as it is a BERT model fine-tuned using scientific text and accomplishes better results in this domain than the base BERT model. Using SciBERT as a base language model saves the time needed to fine-tune a specialized model for scientific text, which would require gathering a number of scientific articles and many time for the fine-tuning process. For the fine-tuning process, the Huggingface’s Transformers library is used, as it implements the needed methods in an intuitive and easy to use API, and is widely supported by the community.

In summary, the framework works following the process:

1. The user inputs scientific texts which are related to their area of study
2. The input is used to fine-tune the SciBERT language model
3. The new, user-customized language model is utilized to assist the user in the writing process

In order to make the fine-tuning process faster and accessible for the end-user, this process is made using Google Colab. A notebook was created for the user to be able to input scientific texts in their area of interest. By running the available scripts, the language model is fine-tuned using the provided texts and the user can download the resultant customized language model for use in the writing process. For the writing assistance, an additional module was developed. In order to develop a quick, multi-platform interface, the module was developed using Python 3 and the *prompt\_toolkit* python library. The

*prompt\_toolkit* library is used for interfacing with the user via terminal, displaying an input text area, while processing the text recommendations in a separate thread to not interrupt the user in the writing process.

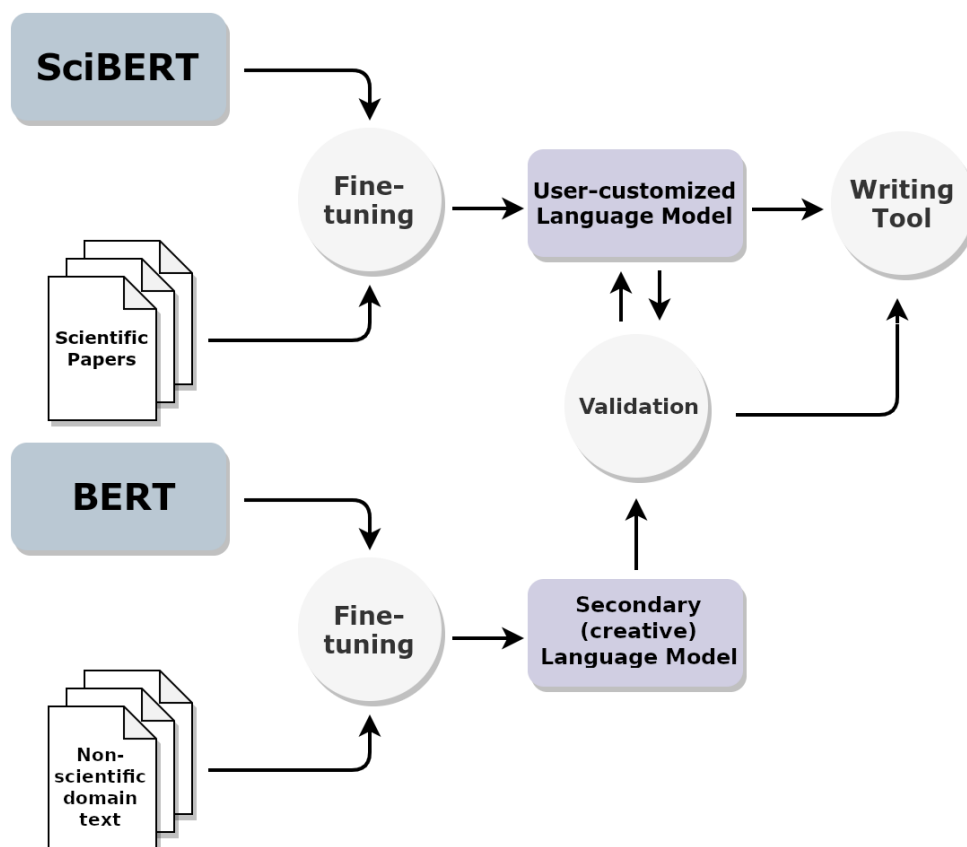
As the user writes their text, the module gathers all text currently in the text area and adds the special token "[MASK]" after the last word. The resulting string is processed using the user-customized language model in a fill-mask task, where the language model returns the top most-appropriate tokens to be placed over the "[MASK]" token. For each one of the five recommendations, a thread is created. Each thread repeats the same procedure for the language model input, but only gets the best rated token repeatedly until the generated string reaches a specific length or is a punctuation mark. Finally, it picks the five best rated next tokens as a 5-way recommendation route and extends each route to generate a better context for the user to pick from. This process is done in order to provide the user with multiple possible routes to continue writing the current sentence.

#### 4.1 Computational Creativity Module

After its initial implementation, a Computational Creativity module is added to NewWriter. The implementation of this module in Newriter works as follows: (1) A secondary Language Model trained with other texts, such as non-scientific texts (eg. literature, news), is used for creative suggestions; (2) Newriter displays variations of the primary Language Model's suggestions alongside the original suggestions.

This new architecture is displayed in Figure 6.

Figure 6 – Newriter’s architecture with added Computational Creativity Module



Source: Author

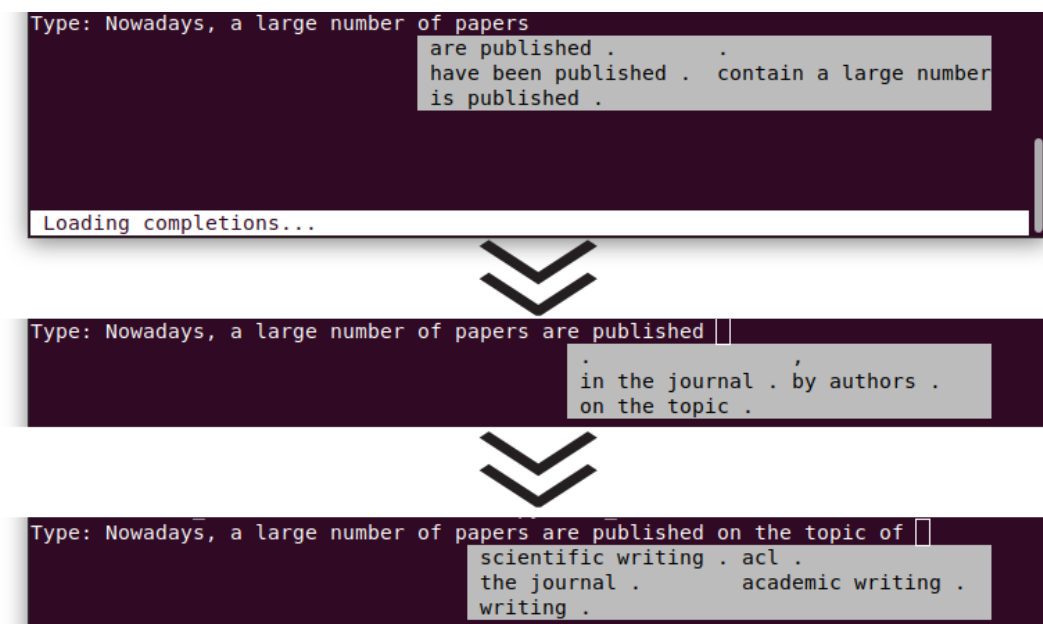
In this new architecture, shown in Figure 6, the Secondary (creative) Language Model is used in the generation of textual recommendations. As one can see, a BERT Language model is fine-tuned using non-scientific domain text. When the writing tool fetches new recommendations, NeWriter returns recommendations generated by the user-customized Language Model. These recommendations are made creative by using the Secondary LM to create variations on the original recommendations. In order to do this, NeWriter generates creative suggestions to change each word in the original recommendation to a new, creative one; and re-evaluates the new recommendations using the original User-customized LM to assure their quality (value). Therefore, the user is presented with creative textual recommendations containing a multitude of creative options to choose from.

In summary, a secondary language model, trained with non-scientific texts, is used along with the main language model to generate creative suggestions as follows:

1. Suggestions are generated by the main Language Model
2. For each suggestion, NeWriter tries to swap each word for other word suggested by the secondary Language Model
3. The text editor software displays the original suggestions along with the best ranked secondary language model's suggestions

Figure 7 shows an example of the writing approach's usage in three sequential moments. As the user writes their text, NEWRITER displays possible writing routes, highly related to the current context. Additionally, the recommended string is highly related to the domain addressed in the scientific texts used for the language model fine-tuning.

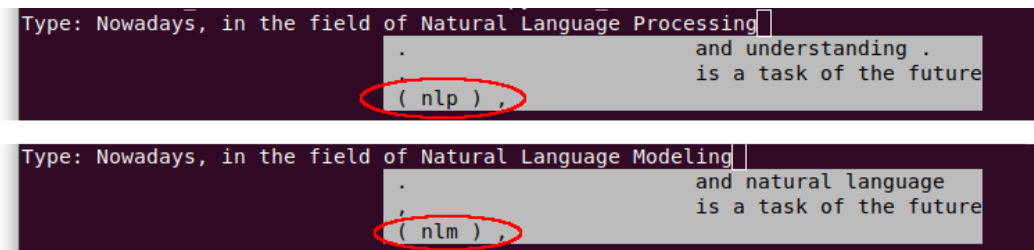
**Figure 7 – Writing software example 1**



**Source: Author**

Figure 8 presents two cases where the language model was capable of memorizing and displaying acronyms for terms used in the natural language area, along with correct parenthesis and punctuation usage.

Figure 8 – Writing software example 2



```
Type: Nowadays, in the field of Natural Language Processing
.
and understanding .
is a task of the future
( nlp ) ,

Type: Nowadays, in the field of Natural Language Modeling
.
and natural language
is a task of the future
( nlm ) ,
```

Source: Author

## 5 METHODOLOGY

In this section, we present experimental setup and tools used to develop and evaluate a scientific text writing assistant. Therefore, Section 5.1 presents the tools that were used for the development of the scientific text writing assistant and Section 5.2 presents the methods and metrics used in this study.

### 5.1 Tools

This study’s objective is to generate recommendations to assist a user in the process of writing scientific text. For this objective’s accomplishment, a base pre-trained language model was selected, a module was developed for the model to be further fine-tuned, and another module was developed for the user to be able to write with the assistance provided by the language model.

For the development of NeWriter we used the HuggingFace’s Transformers library (WOLF et al., 2019). It was created to gather several state-of-the-art language models and architectures into an unified API along with examples, tutorials and scripts, for use by the community. The library contains implementations of state-of-the-art architectures such as BERT, GPT-2, RoBERTa, XLM, DistilBert, among others. There are thousands of pre-trained models available in more than 100 languages with community-contributed models available online. It also has interoperability between PyTorch and TensorFlow 2.0, tools widely used for NLP tasks. The library is highly adopted among both the researcher and practitioner communities.

Other important tool used in the development of this work was Google Colaboratory, also referred simply as Colab. It is a cloud service provided by Google based on Jupyter Notebooks, used for education and research on machine learning. It provides free access to a GPU suitable for deep learning. The research from (PESSOA et al., 2018) shows the service can be used to successfully accelerate not only deep learning applications but also other classes of GPU-centric applications. Experimental results show faster training of a convolutional neural network on Colaboratory’s runtime than using 20 physical cores on a Linux server.

### 5.2 Methods and Metrics

#### 5.2.1 Perplexity

The perplexity metric is commonly used to evaluate the quality of a trained language model. It measures the effectiveness of a probability model, such as a language model, in predicting a given sample. It allows the comparison of language models, with a low perplexity value indicating that a language model is better suited at predicting the given sample. Equation 5.1 shows how perplexity  $p$  is estimated given a probability model  $q$ , for  $N$  test samples and  $b$  is a constant, usually set to 2 (BROWN et al., 1992).

$$p = b^{-\frac{1}{N} \sum_{i=1}^N \log_b q(x_i)} \quad (5.1)$$

#### 5.2.2 User customized language model evaluation

A comparison between SciBERT and a customized language model was made to evaluate the quality of the user-customized language model’s recommendations. A synthetic use-case was used for comparison of the language models, using a 10-fold cross-validation based on their perplexity metric to an input sample. In a k-fold cross-validation, the input sample is equally partitioned into  $k$  sub-samples. For  $k$  iterations,  $k_i$  is used as a

test sample while the other  $k - 1$  parts are used as training samples. Then, the results of the  $k$  tests are averaged for a single estimation result. The most used value for  $k$  is 10, also referred as a 10-fold cross-validation. The selected area of interest for the test was NLP, particularly scientific text writing assistance.

### 5.3 Computational Creativity Module evaluation

In order to evaluate the Computational Creativity model, a second test was made. First, a secondary language model was trained using the base BERT model and trained using 8 classical English literature books, such as Dracula, Frankenstein, and Dr Jekyll and Mr Hyde. For the tests, we used NeWriter with the creativity module, the model trained using classical books, and SciBERT. Then, we used the NLP scientific papers' abstracts as input to collect data. For every test, we started with the first sentence in the paper's abstract to provide a initial context for the language models. For each following word of the abstract, we generated 100 textual suggestions using the three language models, and measured their perplexity using NeWriter. Then, the original following word was added to the text and an additional cycle was executed, as if the text was being wrote as the test was ran.

### 5.4 Value

In order to measure the value of the suggestions made by the Computational Creativity Module, the perplexity metric between base NeWriter and the generated suggestions was used. In doing so, the suggestions had their value validated by the primary Language Model.

### 5.5 Diversity

For the measurement of the diversity of the generated textual recommendations, we collected the suggested words for each language model at every point of the input papers' abstracts. The diversity can be obtained by measuring how many new words NeWriter provides compared to other models. With  $X$  as the unique recommended words by other Language Models and  $Y$  as the unique words recommended by NeWriter, the diversity  $D$  can be calculated as shown in Equation 5.2.

$$D = \{w \in Y | w \notin X\} \quad (5.2)$$

### 5.6 Creativity (RDC)

The final measurement of the textual recommendation's creativity is calculated using the RDC metric, described in Section 2.1. In the scientific text writing domain, we use  $n_a$  as the quotient between new words and the total generated recommendations;  $v_a$  as the quotient between the perplexity and the total vocabulary size; and  $k$  as the quotient between the maximum position embeddings used when accessing the LM and the total vocabulary size.

## 6 EXPERIMENTAL RESULTS

While writing using NeWriter, the displayed recommendations show multiple options for the next word in real-time, and progressively extend upon the recommended writing routes in the following few seconds, without interrupting the user interaction with the text area.

In cases where the recommendations might not be accurate, which can happen specially for recommendation paths starting from a lower score, it does not affect the final user experience, as the final text is curated by the user. Even if the recommendation itself makes sense only to a certain point, it still helps users with idiomatic expressions or connectives to link words together in the text and make the writing process flow better. Those cases happen mostly when there is not much text written to give the LM textual context.

### 6.1 User-customized Language Model evaluation

In Table 1, the 10-fold cross-validation test results are shown. Even with the small provided input, the user-customized Language Model shows to be more than 3% better suited for the task. The training process took, in average, 25 seconds, which is compliant for a one-time pre-processing for using the writing software. The results show that training a customized language model can affect NeWriter’s recommendations to obtain results more accurate to the user’s needs.

**Table 1 – Perplexity comparison between the User-customized LM and SciBERT for specific domain scientific text**

Fold	User LM	SciBERT	Improvement
Fold 1	5.10	5.19	0.02
Fold 2	5.73	5.80	0.01
Fold 3	4.33	4.42	0.02
Fold 4	5.81	5.89	0.01
Fold 5	4.54	4.60	0.01
Fold 6	9.11	9.55	0.05
Fold 7	13.37	13.80	0.03
Fold 8	8.85	9.11	0.03
Fold 9	12.68	13.32	0.05
Fold 10	12.82	13.26	0.03
Average	8.23	8.49	0.03

**Source: Author**

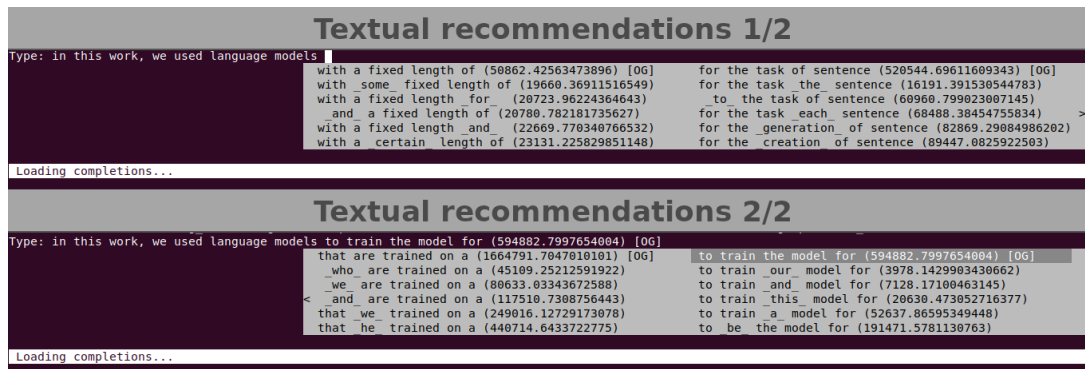
### 6.2 Computational Creativity

As presented in the previous Chapter, we ran tests using input texts’ abstracts simulating their writing process while collecting data from NeWriter, a language model trained using classical literature books, and SciBERT. In this Section, we present value and novelty measures for one of such executions.

With the Computational Creativity module, as one can see in Figure 9, NeWriter displays its original textual recommendations (first item of each column, marked ending with the string "[OG]") along with creative variations (with the creative words marked by underscores before and after them). The recommendations are ordered by their perplexity value, which is shown to the user in parenthesis. The image contains 2 pages of recommendations for the same spot at the beginning of a text writing process.



Figure 9 – Example of creative textual recommendations

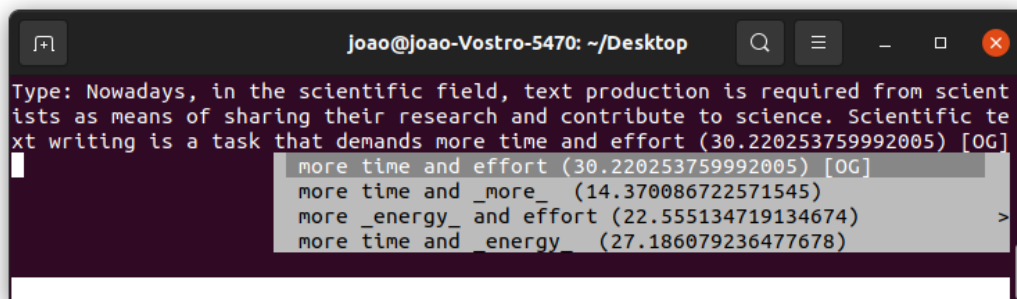


Source: Author

More examples can be seen in Figure 10 and Figure 11. This time, this work’s abstract was used to input more context to the Language Model, which resulted in more accurate results. One can observe that in Figure 10 the recommendations are showing a preference to discuss the same aspect that we discussed in the same spot on the text. However, 11 shows other options that could lead to other discussions, as it goes over the use of computational tools such as NeWriter in the scientific text writing context.

From our tests, we observed that the accuracy of the recommendations improves greatly after the first few sentences. This is expected, as they depend on the context.

Figure 10 – Example of creative textual recommendations in an abstract



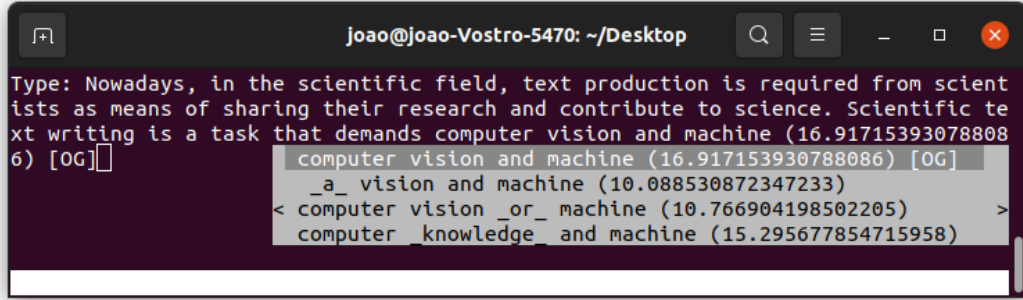
Source: Author

In order to measure the quality of the Creative Module’s textual recommendations, we use base NeWriter as a proxy. For each step of the test, we fetched 100 textual suggestions for each one of the three models: base NeWriter, the language model trained using classical English literature books, and SciBERT. Then, we measured the perplexity of the text with the recommended words using NeWriter to select the 10 best textual recommendations. In doing so, we could compare the average perplexity of each models’ recommendations, as shown in Figure 12.

As one can observe, using NeWriter as means to filter the textual recommendations made by the language model trained using classical English literature books, it was possible to maintain the general quality of its recommendations. In average, the quality of the creativity module’s suggestions were 0.03% and 0.02% worse than the suggestions by NeWriter and SciBERT, respectively.

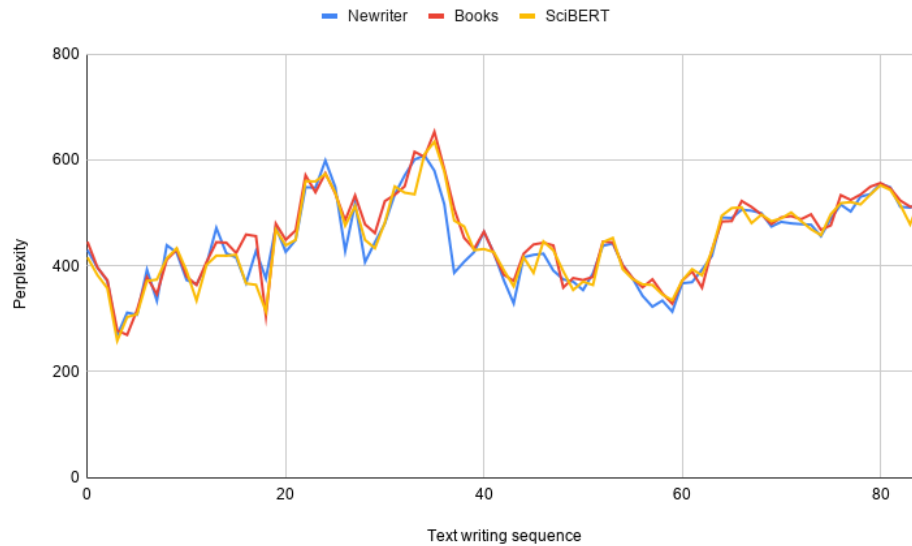
In order to assert the novelty obtained by using the Creativity Model, we measured

Figure 11 – Example of creative textual recommendations in an abstract



Source: Author

Figure 12 – Average perplexity of the textual recommendations for NeWriter, a language model trained using classical English literature books (Books), and SciBERT for each word in a scientific text's abstract



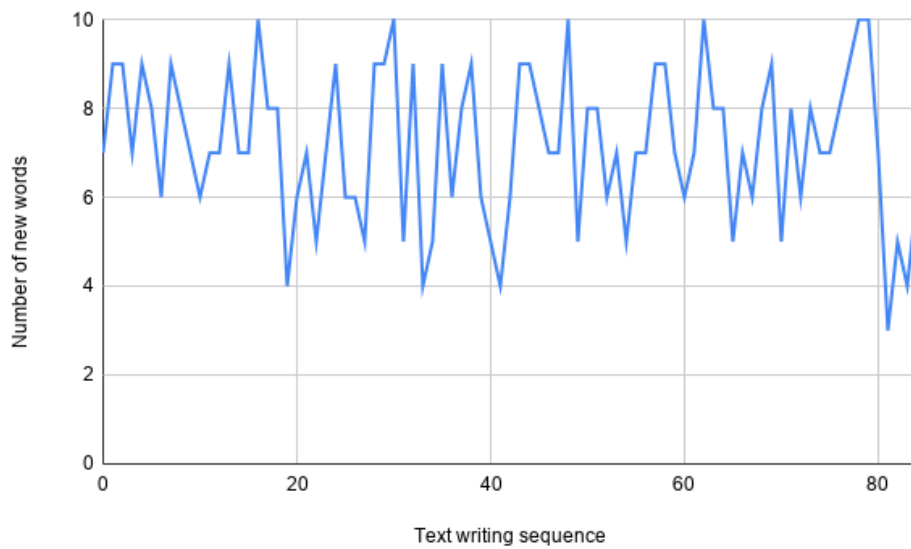
Source: Author

the number of unique words obtained by using the module. For each step of the test, we compared the 10 best rated suggestions between the language model trained using classical English literature books, base NeWriter and SciBERT. The number of words suggested exclusively by the language model trained using classical English literature books was measured and is displayed in Figure 13.

Using the obtained novelty and value results, we calculate this example's RDC using  $n_a = 0.72$ ,  $v_a = 0.98$ , and  $k = 0.016$ , which results in  $RDC = 1.65$ .

The words suggested by NeWriter's creativity module resulted, in average, in 72% more novelty if compared to base NeWriter and SciBERT. These results show that with the creativity module it was possible to generate more diverse textual recommendations compared to the original suggestions, while still maintaining high value.

**Figure 13 – Number of new words the language model trained with classical English literature books suggested comparing its 10 best rated suggestions to base NeWriter and SciBERT, for each word in a scientific text’s abstract**



Source: Author

## 7 CONCLUSION

The scientific text domain can be more welcoming for beginner researchers and for non-native English speakers, and the accomplishments in this research show that the existing NLP tools can be used to reach this objective.

This study’s results show that user-customized language models can be used to improve their effectiveness for scientific text writing assistance compared to state-of-the-art pre-trained models, as shown in the first test suite. We conclude that NEWRITER presents a proof of concept for real-time creative writing assistance using customized, user-trained language models.

As the results show, the quality of the recommendations is maintained, as the perplexity is similar to previous models; while the novelty is obtained as the recommendations contain a higher proportion of new words compared to SciBERT and the original NeWriter. These results mean that NeWriter is successful in using Computational Creativity to expand on the number of terms in its textual recommendations without much impact on the overall quality of the recommendations.

However, there are still more parameters to explore for the language model fine-tuning and other ways the user could be assisted in the writing process.

For future work, this study creates a discussion that can be extended exploring multiple different topics.

In the application aspect, the tool’s interface can be improved for easier usage. The writing software can also be further developed for more tools using the language model’s output. One example is to have recommendations as a selection of words and expressions with higher score in the middle of a sentence, which would also better explore the bidirectionality of the BERT model, as context would be available in both directions. This could greatly improve the Perplexity value for these recommendations.

For the language modelling aspect and for reaching for better results, the language model fine-tuning process can be further tested and asserted: using other example texts, in different writing contexts, using a different topic for the secondary Language Model. The fine-tuning can be made using different numbers of input texts and different values for its parameters in order to compare the impact of those changes in the recommendations generated by the system.

Furthermore, other topic of study that could be further developed is exploring textual automatic correction. In such method, the Language Model could be used as means to correct or improve a full text (draft) written by a beginner researcher in order to create better written text based on the initial version.

Additionally, the tools should be also tested with real users in order to assert their effectivity. A survey can be done with the testers to measure perceived effectivity. Moreover, computed metrics can be collected by the editor program and used to measure the proportion in which the user picks recommended terms, and the proportion of those picks came from the creative Language Model, to better understand the impact of the creative suggestions.



## REFERENCES

- AMMANABROLU, P. et al. *Toward Automated Quest Generation in Text-Adventure Games*. 2020. 14, 17, 24
- ASSAEL, Y.; SOMMERSCHIELD, T.; PRAG, J. Restoring ancient text using deep learning: a case study on greek epigraphy. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, 2019. Disponível em: <<http://dx.doi.org/10.18653/V1/D19-1668>>. 14
- AYE, G. A.; KAISER, G. E. Sequence model design for code completion in the modern ide. *ArXiv*, abs/2004.05249, 2020. 23
- BALDI, P.; ITTI, L. Of bits and wows: A bayesian theory of surprise with applications to attention. *Neural networks : the official journal of the International Neural Network Society*, v. 23 5, p. 649–66, 2010. 17
- BELTAGY, I.; LO, K.; COHAN, A. Scibert: Pretrained language model for scientific text. In: *EMNLP*. [S.l.: s.n.], 2019. 22
- BODEN, M. Computer models of creativity. *AI Magazine*, v. 30, p. 23–34, 07 2009. 14, 17
- BROWN, P. et al. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, v. 18, p. 31–40, 01 1992. 29
- BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. 14
- CHEN, M. X. et al. Gmail smart compose: Real-time assisted writing. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019. (KDD '19), p. 2287–2295. ISBN 9781450362016. Disponível em: <<https://doi.org/10.1145/3292500.3330723>>. 14, 15, 23, 24
- CRUYS, T. Van de. Automatic poetry generation from prosaic text. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020. p. 2471–2480. Disponível em: <<https://www.aclweb.org/anthology/2020.acl-main.223>>. 14, 17, 23, 24
- DAI, Z. et al. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. Disponível em: <<http://arxiv.org/abs/1901.02860>>. 21
- DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019. 21
- DONAHUE, C.; LEE, M.; LIANG, P. Enabling language models to fill in the blanks. *ArXiv*, abs/2005.05339, 2020. 14, 15, 23
- ESTEVA, A. et al. *CO-Search: COVID-19 Information Retrieval with Semantic Search, Question Answering, and Abstractive Summarization*. 2020. 14

- FRANÇA, C. et al. Regent-dependent creativity: A domain independent metric for the assessment of creative artifacts. In: *ICCC*. [S.l.: s.n.], 2016. 17
- GRANGIER, D.; AULI, M. QuickEdit: Editing text & translations by crossing words out. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 272–282. Disponível em: <<https://www.aclweb.org/anthology/N18-1025>>. 24
- ITO, T. et al. *Diamonds in the Rough: Generating Fluent Sentences from Early-Stage Drafts for Academic Writing Assistance*. 2019. 14, 15, 16, 23, 24
- KIEUVONGNGAM, V.; TAN, B.; NIU, Y. *Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2*. 2020. 14
- LI, L.; JIANG, X.; LIU, Q. *Pretrained Language Models for Document-Level Neural Machine Translation*. 2019. 14
- LIU, C. et al. *Jointly Encoding Word Confusion Network and Dialogue Context with BERT for Spoken Language Understanding*. 2020. 14
- MIKOLOV, T.; YIH, W.-t.; ZWEIG, G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, 2013. p. 746–751. Disponível em: <<https://www.aclweb.org/anthology/N13-1090>>. 18
- PESSOA, T. et al. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, PP, p. 1–1, 10 2018. 29
- SANTOS, W. H. et al. Creative culinary recipe generation based on statistical language models. *IEEE Access*, v. 8, p. 146263–146283, 2020. 14, 17, 24
- SHIH, Y.-S.; CHANG, W.-C.; YANG, Y. Xl-editor: Post-editing sentences with xlnet. *ArXiv*, abs/1910.10479, 2019. 23
- UMEHARA, D. *The Will to Keep Winning*. Japan: SHOGAKUKAN INC, 2016. 6
- VASWANI, A. et al. Attention is all you need. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017. p. 5998–6008. Disponível em: <<http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>>. 14, 19, 20, 21
- WANG, A.; CHO, K. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In: *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 30–36. Disponível em: <<https://www.aclweb.org/anthology/W19-2304>>. 16, 21
- WOLF, T. et al. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019. 29