Review

Transformers in the Real World: A Survey on NLP Applications

Narendra Patwardhan, Stefano Marrone * and Carlo Sansone

Department of Electrical Engineering and of Information Technology (DIETI), University of Naples Federico II, Via Claudio 21, 80125 Naples, Italy
* Correspondence: stefano.marrone@unina.it

Abstract: The field of Natural Language Processing (NLP) has undergone a significant transformation with the introduction of Transformers. From the first introduction of this technology in 2017, the use of transformers has become widespread and has had a profound impact on the field of NLP. In this survey, we review the open-access and real-world applications of transformers in NLP, specifically focusing on those where text is the primary modality. Our goal is to provide a comprehensive overview of the current state-of-the-art in the use of transformers in NLP, highlight their strengths and limitations, and identify future directions for research. In this way, we aim to provide valuable insights for both researchers and practitioners in the field of NLP. In addition, we provide a detailed analysis of the various challenges faced in the implementation of transformers in real-world applications, including computational efficiency, interpretability, and ethical considerations. Moreover, we highlight the impact of transformers on the NLP community, including their influence on research and the development of new NLP models.

Keywords: transformers; natural language processing; open-source

check for updates

Citation: Patwardhan, N.; Marrone, S.; Sansone, C. Transformers in the Real World: A Survey on NLP Applications. *Information* **2023**, *14*, 242. https://doi.org/10.3390/info14040242

Academic Editor: Katsuhide Fujita

Received: 24 February 2023
Revised: 5 April 2023
Accepted: 12 April 2023
Published: 17 April 2023

1. Introduction

Natural language processing (NLP) is a field of artificial intelligence that deals with the interaction between computers and humans using natural language. NLP has experienced tremendous growth in recent years due to the increasing availability of large amounts of textual data and the need for more sophisticated and human-like communication between computers and humans. The goal of NLP is to develop algorithms and models that can understand, generate, and manipulate human language in a way that is both accurate and natural. The importance of NLP lies in its ability to transform the way humans and computers interact, enabling more intuitive and human-like communication between them. This has numerous practical applications in areas such as information retrieval, sentiment analysis, machine translation, and question answering, among others. NLP has the potential to revolutionize many industries, such as healthcare, education, and customer service, by enabling more effective and efficient communication and information management. As such, NLP has become an important area of research and development, with significant investment being made in its advancement.

One of the most influential papers in the field of NLP was the first introduction of Transformers [1]. The fundamental unit of said architecture, the transformer block, consists of two main components: a multi-head self-attention mechanism and a fully connected feedforward network. The multi-head self-attention mechanism allows the model to focus on different parts of the input sequence at each layer and weigh the importance of each part in making a prediction. This is accomplished by computing attention scores between each element in the input sequence and all other elements, which are then used to weigh the contribution of each element to the final representation. The multi-head attention mechanism allows the model to learn different attention patterns for different tasks and input sequences, making it more versatile and effective than traditional recurrent or convolutional models. Additionally, the use of self-attention allows the model to process input sequences in parallel, making it much faster and more efficient than sequential models. The feedforward network is essentially a multi-layer perceptron (MLP) that takes in the self-attention-generated representation as input, applies linear transformations with activation functions, and outputs the final representation. This final representation is then passed to the next Transformer block or used for making predictions. Due to operating exclusively on the feature dimension, the transformer block is permutation-equivariant, and as such is especially suitable for sequence processing.

Following the success of transformers in machine translation with the original sequence-to-sequence formulation, researchers began to explore their use in other NLP tasks. One of the most significant developments in this direction was the introduction of BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. [2]. BERT is a pre-trained transformer model that can be fine-tuned for a wide range of NLP tasks, such as sentiment analysis, named entity recognition and question answering. BERT was trained using a contrastive task, where it was asked to predict missing tokens in a sentence given the context of the surrounding tokens. This approach allowed BERT to learn rich contextual representations of words, making it highly effective for a wide range of NLP tasks.

Another development in the field of transformers was the introduction of GPT (Generative Pretrained Transformer) by Radford et al. [3]. GPT is a generative model trained on a large corpus of text with the goal of predicting the next token in a sequence given the context of the surrounding tokens. GPT has been shown to be highly effective for tasks such as text generation, language modeling, and question-answering. Unlike BERT, which was trained using a contrastive task, GPT was trained using a generative task, allowing it to learn a more diverse and complete representation of language.

The success of transformers in NLP can be attributed to several key factors. First, they are highly parallelizable, allowing them to scale effectively to large amounts of data. Second, they are capable of handling variable-length input sequences, making them well-suited to NLP tasks where the length of the input text can vary widely. Moreover, transformers have quickly risen to the top of the leaderboard rankings for most NLP tasks thanks to their effectiveness in capturing long-range dependencies, scalability, and versatility. They have proven to be a powerful tool for NLP, and their continued development and application are likely to have a significant impact on the field.

Transformers have not only revolutionized the field of NLP; they are growing beyond it and finding applications in other areas. For example, transformers have been used in computer vision tasks such as image captioning, where they have been used to generate captions for images based on their content, and have been used in speech recognition, where they have been used to transcribe speech into text.

Another trend in the use of transformers is the development of multimodal models, which allow for the unified modeling and use of text along with other modalities, such as images and audio. These models can learn to understand the relationships between different modalities and can use this understanding to perform a wide range of tasks, such as image-to-text generation, text-to-image generation, and audio-to-text generation. Indeed, transformers are growing beyond the field of NLP and are being used in a wide range of tasks and applications. In addition, they are beginning to allow for the development of multimodal models that can use text as the interface to other modalities and can perform a wide range of tasks in a unified and integrated manner. These developments are likely to have a significant impact on the field and lead to further advancements in the use of transformers for a wide range of tasks.

Unfortunately, many of the successful applications of transformers, such as CoPilot, GPT-3, and ChatGPT, are closed-source, meaning that the underlying algorithms and models are not available for public scrutiny or modification. This can limit the growth and development of the field, as researchers and developers are not able to access the code and models used in these systems and build upon them. Closed development practices raise important ethical questions, particularly in the context of data annotation and the relationship between annotators and model developers. In many cases, annotators who provide the data used to train NLP models do not share a stake in the development or commercialization of the models, and their contributions are often undervalued. For example, Kenyan annotators who were contracted to label data for OpenAI's GPT-3 faced exploitation and poor working conditions, with low pay and long hours. This highlights the need for greater transparency and fairness in the annotation process and the need for annotators to have a stake in the development and commercialization of NLP models. The need for open-source development in the field of transformers is critical for several reasons. First, open-source development allows for greater collaboration and sharing of ideas and knowledge between researchers and developers. This can lead to faster and more effective progress in the field and can help to ensure that the technology is developed in a way that is transparent and ethical. Second, open-source development can help to ensure that the technology is accessible and affordable, especially for researchers and developers in developing countries, who may not have the resources to purchase proprietary software. This can help to democratize access to technology and promote greater innovation and collaboration. Finally, open-source development can help to build trust and credibility in the technology, as the underlying algorithms and models are available for public scrutiny and validation. This can help to ensure that the technology is developed responsibly and ethically and that its impact on society is understood and managed.

The objective of this survey paper is to survey open-source real-world applications of transformers in the field of NLP. We aim to provide a comprehensive overview of the latest developments and trends in the use of transformers for NLP tasks and to highlight the challenges and limitations that need to be addressed. The aim is to provide valuable insights for researchers, developers, and practitioners in the field of NLP, and to help promote further progress and innovation in the use of transformers for NLP tasks.

2. Related Work

The prior research in this field has mainly explored three aspects: structural analysis [4], deep learning architectures and their applications [5], and the utilization of pre-trained models [6].

Moreover, Bender et al. [7] examined the current trend in NLP of developing and deploying larger language models, such as BERT, GPT-2/3, and Switch-C, and questioned whether the size is the only factor driving progress. They provided recommendations for mitigating the risks associated with these models, including considering environmental and financial costs, carefully curating and documenting datasets, evaluating the fit of the technology with research and development goals and stakeholder values, and encouraging research beyond larger language models. Despite serving as a crucial cautionary tale, our survey finds that this work is at odds with the real-world trend of models, which diverge from the limited instances of solely relying on an increased scale.

Other studies such as one by Dang et al. [8], have provided comprehensive overviews of specific NLP tasks such as sentiment analysis, highlighting the promise of deep learning models in solving these challenges by reviewing recent works that construct models based on term frequency–inverse document frequency (TF-IDF) and word embeddings. However, such reviews may overlook concurrent and synergistic advancements by focusing only on one task.

Historically, NLP systems were based on white box techniques, such as rules and decision trees, that are inherently explainable. However, the popularity of deep learning models has led to a decline in interpretability. This lack of transparency in AI systems can erode trust, which is why explainable AI (XAI) has become an important field in AI. Danilevsky et al. [9] focused on XAI works in NLP that have been presented in main NLP conferences in the last seven years, making this the first XAI survey specifically focused on the NLP domain.

Deep learning models require large amounts of data, which can be a challenge for many NLP tasks, especially for low-resource languages. Additionally, these models require significant computing resources. The increasing demand for transfer learning is driven by the need to overcome these limitations and make the most of the large trained models that are emerging. Alyafeai et al. [10] have examined the recent developments in transfer learning in the field of NLP.

In contrast to prior work, we seek to explore how Transformer-based models can be effectively applied in practical scenarios, particularly when source code is accessible. By concentrating on this specific architectural family and its practical implementations, our investigation intends to provide deeper insights into the benefits and limitations of Transformers as well as into potential areas for further innovation and improvement in natural language processing. Another related study by Wu et al. [11] delved into the application of graph neural networks (GNNs) for NLP tasks. GNNs share similarities with Transformers in their ability to capture long-range dependencies and complex relationships between data entities. However, the unique formulation of GNNs sets them apart from traditional Transformers, warranting separate investigation, as GNNs explicitly model data as graphs and leverage graph structures to perform computation and information propagation, whereas transformers work directly on flattened sequences, and as such are more suitable for processing language data.

3. Methodology

There are several databases available on the internet that are focused especially on collecting scientific papers, with differences in terms of the type, amount, and detail of reported information. While the reasons for this variety are different, the primary one is because of the target audience (e.g., practitioners, researchers, students, etc.). Among these databases, PapersWithCode is a valuable source for current advancements and trends in the NLP field, as it is totally built around the idea of collecting only papers for which a working code repository is available that can be used to successfully reproduce the claimed results. In this paper, we focus on real-world applications; thus, an initial corpus of applications was compiled by extracting natural language processing (NLP) tasks from the PapersWithCode website. Our initial list consisted of 572 entries, which underwent several heuristic-based filtering steps to ensure conciseness, accuracy of match, and relevance in real-world contexts. In particular:

- We first eliminated entries with a high degree of similarity, which we defined as a fuzzy similarity ratio greater than 95 compared to other entries in the list. We calculated the fuzzy similarity ratio using the FuzzyWuzzy library, which measures the similarity between two strings based on the Levenshtein distance which represents the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. The formula used to compute the fuzzy similarity ratio is presented in Equation (1). This step targeted "shorter" entries, i.e., those with fewer characters, in order to maximize information content in cases of high overlap.

$$F(string1, string2) = (1 - \frac{D_{Levenshtein}}{max(len(string1), len(string2))}) * 100 \tag{1}$$

- Second, we removed entries composed solely of uppercase letters, as they were likely abbreviations or acronyms unrelated to our target applications.
- Third, entries with fewer than five characters were excluded, as they were potentially incomplete or illegible.
- Fourth, we discarded entries containing numbers, as these were more likely to represent numerical codes or identifiers than descriptive labels.
- Finally, we removed entries with more than five words in order to exclude overly specific tasks not pertinent to the broader application themes that we aimed to investigate.

After applying these heuristics, we conducted manual filtering to eliminate benchmark-specific applications and to group similar tasks. In order to locate relevant papers, we searched Google Scholar using the refined application list as keywords, limiting the search to articles published within the last five years and sorting by relevance. This time, the frame was chosen based on the release of the original transformers paper in 2017. We further narrowed the results to papers with at least one citation, ensuring relevance and recency.

Additionally, we excluded papers by parsing for Git provider URLs, such as GitHub or GitLab, embedded within the papers. We then cross-referenced the repository's title and description against the paper's title, discarding papers where the repository title did not match the paper title. This step ensured that the retained papers had a strong connection to their corresponding repositories, further refining the quality and relevance of our corpus.

For each application, we conducted a manual review of the identified papers to thoroughly examine the methodology, results, and contributions and to critically evaluate the limitations and challenges associated with each application. The findings from this review furnished a comprehensive overview of the latest developments and trends in transformer usage for NLP tasks.

3.1. Categorization

Here, we present a schema with two primary factors to categorize NLP applications, namely, the degree of modality and the kind of transformer architecture. Unimodal NLP applications deal with a single modality, such as text or speech, while multimodal NLP applications deal with multiple modalities, such as text, speech, and images. Text often serves as the primary interface in multimodal applications. The kind of transformer architecture used in NLP applications plays a crucial role in determining the overall performance of the system. Encoder-only transformers are used for discriminative tasks such as sentiment analysis and named entity recognition, while decoder-only transformers are used for tasks such as text generation and summarization. Encoder–decoder transformers are used for tasks such as machine translation and image captioning. Understanding the degree of modality and kind of transformer architecture used in NLP applications is important for choosing the right approach to a given task.

4. Applications

In this section, we provide a comprehensive overview of the primary application groups within the degree-of-modality-based categorization framework outlined in Section 3.1. Our aim is to provide a clear understanding of the common architectural choices made for each application group. To enhance the comprehension of the technical details, we present the links to the source code at the end of the discussion, which encompasses each task subcategory. We hope this approach will allow the reader to gain a deeper understanding of the implementation of the described concepts and to explore the code in greater detail if desired.

4.1. Unimodal Applications

Unimodal applications refer to AI-based systems that primarily focus on processing and analyzing text as their main modality. In the subsequent sections, we delve into the primary categories of unimodal applications, which include Language Modeling (Section 4.1.1), Question Answering (Section 4.1.2), Machine Translation (Section 4.1.3), Text Classification (Section 4.1.4), Text Generation (Section 4.1.5), Text Summarization (Section 4.1.6), Sentiment Analysis (Section 4.1.7), Named Entity Recognition (Section 4.1.8), and Information Retrieval (Section 4.1.9). These categories exemplify the diverse range of applications and capabilities that AI systems can achieve by focusing on text-based information.

4.1.1. Language Modeling

Language modeling is a fundamental task in NLP that involves predicting the next word in a sequence of text based on the preceding words. The goal of language modeling is to estimate the probability distribution of sequences of words in a given language, and is used as a building block for many NLP tasks such as machine translation, speech recognition, and text generation. Language modeling can be easily extended to more complex NLP tasks such as sentence-pair modeling, cross-document language modeling, and definition modeling. By leveraging the knowledge learned from language modeling, these tasks can benefit from improved accuracy and efficiency. Language modeling typically follows the decoder-only architecture popularized by the GPT family [3,12,13].

However, when used for generation, language modeling is often limited in its ability to handle complex language phenomena, and the large size of models makes them computationally expensive to fine-tune for specific tasks. Typically, these models are utilized by providing additional context in the form of a prompt (called "prompt tuning"), either manually or through automated selection [14]. Transformer-XL [15] is a unique neural architecture intended for language modeling that can learn relationships beyond a set length while keeping temporal consistency. It has a segment-level recurrence mechanism and an innovative positional encoding scheme that captures longer-term dependencies while addressing context fragmentation. As a result, Transformer-XL outperforms both LSTMS and standard transformers on both short and long sequences, and is significantly faster during evaluation.

Dynamic evaluation enhances models by adapting to recent sequence history through gradient descent, capitalizing on recurrent sequential patterns. It can exploit long-range dependencies in natural language, such as style and word usage. Krause et al. [16] investigated the benefits of applying dynamic evaluation to transformers, aiming to determine whether transformers can fully adapt to recent sequence history. Their work builds on top of the previously mentioned Transformer-XL model.

Recently, a promising new direction in language modeling involves the use of reinforcement learning (RL)-based fine-tuning. In this approach, a pre-trained language model is fine-tuned using RL to optimize a particular task-specific reward function [17]. This allows the model to learn from its predictions, leading to improved accuracy and generalization performance on the target task. Additionally, fine-tuning with RL can be accomplished with much smaller models, making it computationally more efficient and faster. This new direction of RL-based fine-tuning has shown promising results on a variety of NLP tasks and is a promising avenue for further research and development in the field of language modeling. The relevant source code repositories for the papers discussed in this section can be found in Table 1.

Table 1. Source code links for Language Modeling.

Paper Title	**Link to Source**
Autoprompt: Eliciting knowledge from language models with automatically generated prompts	https://github.com/ucinlp/autoprompt, accessed on 1 April 2023.
Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context	https://github.com/kimiyoung/transformer-xl, accessed on 1 April 2023.
Dynamic Evaluation of Transformer Language Models	https://github.com/benkrause/dynamiceval-transformer, accessed on 1 April 2023.

4.1.2. Question Answering

Question answering is a task in NLP that involves automatically answering questions posed in natural language. The goal of question answering is to extract the relevant information from a given text corpus and present it as an answer to a user's question. Question-answering systems can operate over a wide range of text types, including news articles, Wikipedia pages, and others, and can be designed to answer a wide range of questions, including fact-based questions, opinion questions, and others. There are several subtasks within QA, each with its unique challenges and requirements. Among the most common subtasks are:

- **Open-Domain Question Answering (ODQA)**: This task involves finding an answer to a question from an open domain, such as the entire internet or a large corpus of text. The goal is to find the most relevant information to answer the question, even if it requires synthesizing information from multiple sources. Reformer, introduced by Kitaev et al. [18], has been shown to excel at ODQA, with its success attributed to the use of locality-sensitive hashing which enables far larger context windows than ordinary transformers.

- **Conversational Question Answering (CQA)**: This task involves answering questions in a conversational setting, where the model must understand the context of the conversation and generate an answer that is relevant and appropriate for the current conversational context. SDNet [19] utilizes both inter-attention and self-attention mechanisms to effectively process context and questions separately and fuse them at multiple intervals.
- **Answer Selection**: This task involves ranking a set of candidate answers for a given question, where the goal is to select the most accurate answer from the candidate set. Fine-tuning pre-trained transformers has been shown to be an effective method within answer selection [20].
- **Machine Reading Comprehension (MRC)**: This task involves understanding and answering questions about a given passage of text. The model must be able to comprehend the text, extract relevant information, and generate an answer that is accurate and relevant to the question. XLNet [21] uses a permutation-based training procedure that allows it to take into account all possible combinations of input tokens, rather than just the left-to-right order as in traditional transformer models. XLNet's ability to capture long-range dependencies and its strong pre-training make it a highly competitive model for the MRC task.

The source code links for the papers discussed in this section can be found in Table 2.

Table 2. Source code links for Question Answering.

Paper Title	Link to Source
Reformer: The efficient transformer	https://github.com/google/trax/tree/master/trax/models/reformer, accessed on 20 March 2023.
Sdnet: Contextualized attention-based deep network for conversational question answering	https://github.com/Microsoft/SDNet, accessed on 20 March 2023.
Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection	https://github.com/alexa/wqa_tanda, accessed on 20 March 2023.
Xlnet: Generalized autoregressive pretraining for language understanding	https://github.com/zihangdai/xlnet, accessed on 20 March 2023.

4.1.3. Machine Translation

Machine translation (MT) is the task of automatically converting a source text in one language to a target text in another language. The goal of machine translation is to produce a fluent and accurate translation that conveys the meaning of the source text in the target language. MT models often follow an encoder–decoder architecture to capture the context effectively using bidirectional encoder and be able to generate text of arbitrary length, following the original formulation of transformer architecture [1]. There are several subtasks within MT, each with its unique challenges and requirements. Among the most common subtasks are:

- **Transliteration**: This task involves translating text from one script to another, such as translating between the Latin and Cyrillic scripts. Transliteration is different from traditional MT in that it typically involves preserving the meaning of words, rather than translating the meaning of words to another language. Because transliteration requires support for non-Latin or often arbitrary characters, models developed for it often use character-level or byte-level encoding. For instance, a model by Wu et al. [22] based on the same principles has shown strong performance on transliteration, outperforming established recurrent baselines with large batch sizes.

- **Unsupervised Machine Translation (UMT)**: This task involves translating between two languages without any parallel training data, meaning that there is no corresponding text in the target language for the source language text. UMT models are typically trained on monolingual data in each language and use various unsupervised techniques to learn the relationship between the languages. Zhu et al. [23] show strong performance on English–French and English–Romanian pairs without any fine-tuning. The idea behind the BERT-fused approach is to leverage pre-training with BERT to better understand the relationships between languages and to use the sequence-to-sequence architecture to generate translations;
- **Bilingual Lexicon Induction (BLI)**: This task involves inducing a bilingual lexicon, which is a mapping between words in two languages. BLI is a critical component of MT, and is often used as a pre-processing step to generate initial alignments between words in the source and target languages.

The source code for the papers discussed in this section can be found in Table 3, as referenced.

Table 3. Source code links for Machine Translation.

Paper Title	Link to Source
Applying the transformer to character-level transduction	https://github.com/shijie-wu/neural-transducer, assessed on 1 April 2023.
Incorporating BERT into neural machine translation	https://github.com/bert-nmt/bert-nmt, assessed on 1 April 2023.

4.1.4. Text Classification

Text classification is the task of categorizing a text into one or more predefined categories based on its content. The goal of text classification is to automatically assign a label to a given text based on its content, allowing it to be organized and categorized for easier analysis and management. These models are trained on annotated text data in order to learn the relationship between the text content and its label, and can then be used to classify new unseen text data. Text classification models typically follow a decoder-only architecture in order to effectively capture the entirety of context using bidirectional attention. While text classification is often the most varied use case due to its commercial importance, two primary subcategories are prominent:

- **Document Classification**: This task involves assigning a label or category to a full document, such as a news article, blog post, or scientific paper. Document classification is typically accomplished by first representing the document as a numerical vector and then using a machine learning model to make a prediction based on the document's representation. LinkBERT [24] extends the pre-training objective of BERT to incorporate links between documents, which results in better classification quality.
- **Cause and Effect Classification**: This task involves identifying the cause and effect relationship between two events described in a sentence or paragraph. An approach by Hosseini et al. [25] has shown the efficacy of the language modeling paradigm by verbalizing knowledge graphs and using them as a pre-training corpus for a language model. The model obtains acceptable performance without any further fine-tuning or prompting.

The corresponding source code for the papers reviewed in this section can be found in Table 4.

Table 4. Source code links for Text Classification.

Paper Title	Link to Source
Linkbert: Pretraining language models with document links	https://github.com/michiyasunaga/LinkBERT, accessed on 20 March 2023.
Knowledge-augmented language models for cause-effect relation classification	https://github.com/phosseini/causal-reasoning, accessed on 20 March 2023.

4.1.5. Text Generation

Text Generation is a task in NLP in which the objective is to produce new text automatically, typically starting from a given prompt or input. The output can be a single word, phrase, sentence, or full-length piece of text, and is used for chatbots, content creation, and more. The generated text should reflect an understanding of the input and the language being generated, and the quality and coherence of the generated text can vary depending on the approach used. Text generation has seen a surge of interest following the release of commercial APIs such as GPT, Cohere, and ChatGPT. Text generation typically follows a decoder-only architecture; however, recent issues with prompt-injection attacks have migrated part of the focus towards encoder = -decoder models that have been instruction-tuned, such as T5 [26]. While the most prominent approach to text generation is based on prompting, several other forms of generation have been studied in the literature and have found commercial success as well. Text generation subtasks include:

- **Dialogue Generation**: This category focuses on generating text in the form of a conversation between two or more agents. Dialogue generation systems are used in various applications, such as chatbots, virtual assistants, and conversational AI systems. These systems use dialogue history, user input, and context to generate appropriate and coherent responses. P2-BOT [27] is a transmitter–receiver-based framework that aims to explicitly model understanding in chat dialogue systems through mutual persona perception, resulting in improved personalized dialogue generation based on both automated metrics and human evaluation.
- **Code Generation**: This category focuses on generating code based on a given input, such as a natural language description of a software problem. Code generation systems are used in software development to automate repetitive tasks, improve productivity, and reduce errors. These systems can be trained to use expert knowledge, and can be specialized for a single programming language, such as SQL [28], or trained on a large corpus to support various programming languages and different programming paradigms [29];
- **Data-to-Text Generation**: This category focuses on generating natural language text from structured data such as tables, databases, or graphs. Data-to-text generation systems are used in various applications, such as news reporting, data visualization, and technical writing. These systems use natural language generation techniques to convert data into human-readable text, taking into account the context, target audience, and purpose of the text. Control Prefixes [30] extend prefix tuning by incorporating input-dependent information into a pre-trained transformer through attribute-level learnable representations, resulting in a parameter-efficient data-to-text model.

Table 5 contains links to the source code corresponding to the papers discussed in this section.

Table 5. Source code links for Text Generation.

Paper Title	Link to Source
Exploring the limits of transfer learning with a unified text-to-text transformer	https://github.com/google-research/text-to-text-transfer-transformer, accessed on 2 March 2023.
You impress me: Dialogue generation via mutual persona perception	https://github.com/SivilTaram/Persona-Dialogue-Generation, accessed on 2 March 2023.
Content Enhanced BERT-based Text-to-SQL Generation	https://github.com/guotong1988/NL2SQL-RULE, accessed on 2 March 2023.
Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation	https://github.com/salesforce/CodeT5, accessed on 2 March 2023.
Control prefixes for parameter-efficient text generation	https://github.com/jordiclive/ControlPrefixes, accessed on 2 March 2023.

4.1.6. Text Summarization

Text Summarization is a task in NLP where the goal is to condense a given text into a shorter and more concise version while preserving its essential information. This is typically accomplished by identifying and extracting the most important information, sentences, or phrases from the original text. The resulting summary can be a few sentences long or a single bullet point, and is intended to provide a quick overview of the content without the need to read the entire text. Text summarization is used in a variety of applications, such as news aggregation, document summarization, and more. Text summarization typically requires an encoder–decoder architecture to completely capture the source information. Depending on the input size, standard attention might prove too costly due to the quadratic computation cost based on the sequence length. Methods such as [31] replace the attention layer with an equivalent (here, a pooled attention module) to efficiently handle larger context windows. Under this category, possible tasks are as follows:

- **Extractive Summarization**: This is the most straightforward subtask of text summarization, where the goal is to extract the most important sentences or phrases from a document and present them as a summary. Extractive summarization methods typically use a combination of information retrieval and natural language processing techniques to identify the most informative sentences or phrases in a document.
- The attention mechanism of Longformer [32] is a substitute for standard self-attention, and merges localized windowed attention with globally focused attention. The encoder–decoder version of the longformer (called LED) has demonstrated its effectiveness on the arXiv summarization dataset, and is used often for processing long contexts in real-world applications.
- **Abstractive Summarization**: This subtask aims to generate a summary by synthesizing new information based on the input document. Abstractive summarization methods typically use deep learning models, such as recurrent neural networks or transformers, to generate a summary. These models are trained on large amounts of data and can generate summaries that are more concise and coherent than extractive summaries. mBart is a sequence-to-sequence transformer [33] trained on multiple large-scale monolingual corpora with the objective of denoising. Due to its rich pre-training dataset and ability to process multiple languages using the same network, it excels at abstractive summarization.
- **Multi-Document Summarization**: This subtask addresses the problem of summarizing multiple related documents into a single summary. Multi-document summarization methods typically use information retrieval techniques to identify the most important documents and natural language processing techniques to generate a summary from the selected documents. While prior state-of-the-art methods relied on

GNNs to take advantage of inherent connectivity, Primer by Xiao et al. [34] has shown better performance in zero-shot, few-shot, and fine-tuned paradigms by introducing a new pretraining objective in the form of predicting masked salient sentences.

- **Query-Focused Summarization**: This subtask focuses on summarizing a document based on a specific query or topic. Query-focused summarization methods typically use information retrieval techniques to identify the most relevant sentences or phrases in a document and present them as a summary. Baumel et al. [35] introduced a pre-inference step involving computing the relevance between the query and each sentence of the document. The quality of summarization has been shown to improve when incorporating this form of relevance as an additional input. Support for multiple documents is achieved using a simple iterative scheme that uses maximum word count as a budget.
- **Sentence Compression**: This subtask focuses on reducing the length of a sentence while preserving its meaning. Sentence compression methods typically use natural language processing techniques to identify redundant or unnecessary words or phrases in a sentence and remove them to create a more concise sentence. Ghalandari et al. [36] trained six-layer model called DistilRoBERTa with reinforcement learning to predict a binary classifier that keeps or discards words to reduce the sentence length.

The source code for the papers discussed in this section can be accessed via the links provided in Table 6.

Table 6. Source code links for Text Summarization.

Paper Title	**Link to Source**
Adapting Pretrained Text-to-Text Models for Long Text Sequences	https://github.com/facebookresearch/bart_ls, accessed on 10 March 2023.
Longformer: The long-document transformer	https://github.com/allenai/longformer, accessed on 10 March 2023.
Multilingual denoising pre-training for neural machine translation	https://github.com/facebookresearch/fairseq/tree/main/examples/mbart, accessed on 10 March 2023.
Primer: Pyramid-based masked sentence pre-training for multi-document summarization	https://github.com/allenai/PRIMER, accessed on 10 March 2023.
Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models	https://github.com/talbaumel/RSAsummarization, accessed on 10 March 2023.
Efficient Unsupervised Sentence Compression by Fine-tuning Transformers with Reinforcement Learning	https://github.com/complementizer/rl-sentence-compression, accessed on 10 March 2023.

4.1.7. Sentiment Analysis

Sentiment Analysis is a task in NLP with the goal of determining the sentiment expressed in a given text. This is typically accomplished by assigning a sentiment label such as positive, negative, or neutral to the text based on its contents. The sentiment can be expressed in different forms, such as opinions, emotions, or evaluations, and can be expressed at various levels of granularity, such as at the document, sentence, or aspect level. Sentiment Analysis is used in a variety of applications, such as customer service, marketing, and opinion mining. The quality of the sentiment analysis results can be influenced by factors such as the subjectivity of the text, the tone, and the context in which the sentiment is expressed. Instruction-tuned models such as T5 [26] are often used in a zero-shot manner to perform sentiment analysis. XLNet [21] has been shown to be effective on several sentiment analysis leaderboards such as SST-2, IMDB, and Yelp fine-grained. The source code for the papers discussed in this section can be found in Table 7, where links to the repositories are provided.

Table 7. Source code links for Sentiment Analysis.

Paper Title	Link to Source
Exploring the limits of transfer learning with a unified text-to-text transformer	https://github.com/google-research/text-to-text-transfer-transformer, accessed on 1 March 2023.
Xlnet: Generalized autoregressive pretraining for language understanding	https://github.com/zihangdai/xlnet, accessed on 1 March 2023.

4.1.8. Named Entity Recognition

Named Entity Recognition (NER) is a task in NLP with the goal of identifying and categorizing named entities present in a given text into predefined categories such as person names, organizations, locations, dates, and more. NER is an important subtask of information extraction, and is used as an intermediate step in various applications such as question-answering, event extraction, and information retrieval. The quality of NER results can be influenced by factors such as the ambiguity of entity names, the presence of entity mentions with different forms, and the context in which the entities are expressed.

NER systems typically use machine learning techniques such as supervised learning to learn and identify named entities based on annotated training data. The output of NER is usually a sequence of tagged words, with each word being labeled with its corresponding entity class. As such, it falls under the paradigm of token-wise classification, with the added caveat that unlike most classification tasks it includes a null category. As the sentence and output lengths in NER are equal, it typically utilizes an encoder-only architecture. While the approach of fine-tuning a pretrained model with a classification head added on top for NER works well in practice, Automated Concatenation of Embeddings (ACE) [37] has shown improved results using an ensemble of several pretrained models while training only a simple classifier on top using reinforcement learning. The relevant source code repository for ACE can be found in Table 8.

Table 8. Source code links for Named Entity Recognition.

Paper Title	Link to Source
Automated concatenation of embeddings for structured prediction	https://github.com/Alibaba-NLP/AC, accessed on 1 March 2023.

4.1.9. Information Retrieval

Information Retrieval (IR) is a task in NLP with the goal of retrieving relevant information from a large collection of documents in response to a user query. This is typically accomplished by matching the query terms against the document content and ranking the documents based on their relevance to the query. IR systems can be used for various applications, such as web search, document search, and question answering. The quality of the retrieval results can be influenced by factors such as the relevance of the documents, the effectiveness of the ranking algorithm, and the representation of the documents and queries.

IR systems are typically classified further based on the level of granularity, such as document, paragraph, sentence, etc. While symbolic methods dominated IR leaderboards for a long time, transformer-based embeddings are quickly becoming the norm within the research community. Commercial use however remains in its infancy due to more demanding hardware requirements as compared to symbolic methods. The typical methods for retrieval include the use of a pretrained model such as RoBERTa [38] in a Siamese fashion to find the similarity between two embeddings. For larger datasets, the embeddings are precomputed and stored in a vector database for faster lookup. The source code link for RoBERTa can be found in Table 9.

Table 9. Source code links for Information Retrieval.

Paper Title	**Link to Source**
RoBERTa: A Robustly Optimized BERT Pretraining Approach	https://github.com/facebookresearch/fairseq, accessed on 1 March 2023.

4.2. Multimodal Applications

Multimodal applications are AI-driven systems that leverage multiple modalities, such as text, images, and videos, to process and analyze information. By integrating various forms of data, these applications enable more comprehensive and versatile solutions in diverse domains. In the subsequent sections, we explore the primary categories of multimodal applications, including Generative Control (Section 4.2.1), Description Generation (Section 4.2.2), and Multimodal Question Answering (Section 4.2.3). These categories showcase the potential of AI systems to deliver more robust and context-aware insights by utilizing different data types, ultimately leading to improved performance and user experience across a wide array of applications.

4.2.1. Generative Control

Generative Control is a task in multimodal NLP in which text is used as an interface to generate another modality, such as images or speech. The goal of Generative Control is to generate a target modality that corresponds to a given text description or instruction. For example, based on a textual description of an object, such as "a red sports car," the task of Generative Control would be to generate an image of a red sports car. Generative Control combines the strengths of NLP and computer graphics or speech synthesis to produce high-quality and semantically meaningful outputs in the target modality. It has applications in areas such as computer vision, robotics, and human–computer interaction. Rombach et al. [39] used text as the primary modality for image generation. An open-source implementation of this method named StableDiffusion has generated vast interest as an alternative to the commercial API based on prior work by Ramesh et al. [40]. In the domain of text-to-speech (TTS), Wang et al. [41] combined traditional neural codecs with transformers, which is able to outperform zero-shot TTS systems by treating the problem as conditional language modeling.

The source code for the papers discussed in this section can be found in Table 10, as referenced.

Table 10. Source code links for Generative Control.

Paper Title	**Link to Source**
High-resolution image synthesis with latent diffusion models	https://github.com/CompVis/latent-diffusion, accessed on 1 March 2023.
Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers	https://github.com/microsoft/unilm/tree/master/valle, accessed on 1 March 2023.

4.2.2. Description Generation

Description Generation is a task in which text is generated to describe another modality, such as an image or a point cloud. The goal of Description Generation is to automatically produce a textual description of the content of the target modality that accurately captures its key aspects and characteristics. For example, given an image of a scene, the task of Description Generation would be to generate a textual description of the objects, actions, and attributes present in the scene. Description generation commonly includes tasks such as image captioning and scene understanding.

mPLUG [42] is a new transformer-based vision-language model that combines cross-modal understanding and generation, achieving state-of-the-art results on various vision-language tasks and addressing the inefficiency and linguistic signal issues in existing models through its efficient cross-modal skip-connections. The link to the corresponding source code for mPlug can be found in Table 11.

Table 11. Source code links for Description Generation.

Paper Title	Link to Source
mPLUG: Effective and Efficient Vision-Language Learning by Cross-modal Skip-connections	https://github.com/alibaba/AliceMind/tree/main/mPLUG, accessed on 10 March 2023.

4.2.3. Multimodal Question Answering

Multimodal Question Answering (QA) is a task with the goal of answering questions about a given multimodal input, such as an image or a video, using information from multiple modalities. The task involves combining information from text, images, audio, and other modalities to accurately answer questions about the content of the input. For example, given an image of a scene and a question about the scene, such as "what is the color of the car?", the task of Multimodal QA would be to identify the car in the image and answer the question with the correct color. Multimodal QA requires the integration of NLP, computer vision, and other relevant modalities to accurately answer questions about the content of the input. It has applications in areas such as intelligent tutoring systems, customer service, and multimedia retrieval.

Models utilized for multimodal QA usually show heterogeneity to effectively process modalities other than text. For example, Plepi et al. [43] used a stacked pointer network to aggregate information from a knowledge graph for conversational question answering. Unik-qa [44] uses a retriever–reader architecture that fetches the most relevant documents related to the question based on dense embedding similarity and uses it as context during generation, supporting multiple modalities such as text, tables, lists, and knowledge bases within the documents.

BEiT [45] performs masked language modeling on images, texts, and image-text pairs using a shared backbone. For visual question answering, the model utilizes a fusion encoder in which patch and word embeddings share attention with the attention component of the transformer block while having separate feed-forward layers in the initial stages. By simply fine-tuning a classifier on top, BEiT outperforms all previous methods by a large margin.

Table 12 contains links to the source code corresponding to the papers discussed in this section.

Table 12. Source code links for Multimodal Question Answering.

Paper Title	Link to Source
Context transformer with stacked pointer networks for conversational question answering over knowledge graphs	https://github.com/endrikacupaj/CARTON, accessed on 10 March 2023.
Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering	https://github.com/facebookresearch/UniK-QA, accessed on 10 March 2023.
Image as a foreign language: Beit pretraining for all vision and vision-language tasks	https://github.com/microsoft/unilm/tree/master/beit, accessed on 10 March 2023.

5. Discussion and Conclusions

In conclusion, this comprehensive work has delved into the landscape of transformers in real-world NLP applications, specifically focusing on those with open-source implementations. Through our extensive survey, we have identified and examined twelve main categories and seventeen subcategories of tasks that showcase the versatility and power of transformer models in addressing various challenges in the field of natural language processing.

Our decision to focus on papers with accompanying open-source implementations was driven by the aim of promoting accessibility, reproducibility, and collaboration within the research community. Open-source implementations facilitate a more transparent and inclusive environment, enabling researchers and practitioners from various backgrounds to build upon, refine, and adapt existing models and techniques. This approach encourages innovation by lowering the barriers to entry and fostering a more diverse range of perspectives. Additionally, emphasizing reproducibility can ensure that research findings are reliable and robust, ultimately contributing to the overall advancement and credibility of the field.

In all, the field of natural language processing (NLP) has witnessed significant advancements thanks to the introduction of various transformer architectures, which have proven effective in addressing long-range dependencies, scalability, and versatility across various tasks. While research tends to focus on larger models with increasing parameter counts, real-world applications often rely on models with fewer than one billion parameters. As demonstrated in Figure 1, smaller models can provide comparable or even superior performance, particularly when training data are limited. Moreover, more compact models are faster to train, simpler to deploy, and offer enhanced interpretability, which is crucial for understanding the rationale behind their predictions in practical settings. Consequently, the pursuit of increasingly large models in research should be balanced with an appreciation for the potential advantages of smaller models in real-world applications.

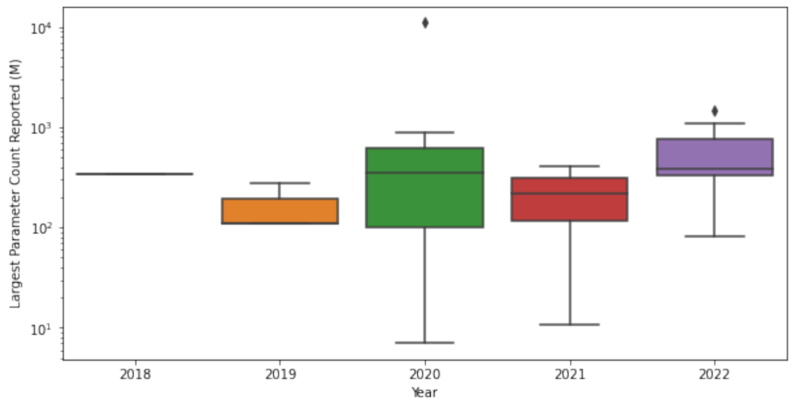

Figure 1. Reported parameter count of the largest model by year. The grey diamonds represent outliers.

This analysis contributes to our understanding of the strengths and weaknesses of transformers in practice, and offers valuable guidance for subsequent research in this domain. By showcasing the potential of open-source development in transformer-based models, we hope to encourage further collaboration, innovation, and ethical considerations in NLP research and application.

Author Contributions: Conceptualization, N.P. and S.M.; methodology, N.P. and S.M.; software, N.P.; validation, all; investigation, N.P.; resources, N.P.; data curation, N.P.; writing—original draft preparation, N.P.; writing—review and editing, S.M. and C.S.; visualization, N.P.; supervision, S.M. and C.S.; project administration, S.M. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was made possible thanks to the generous support of the SIMAR GROUP s.r.l., Monte Urano (FM, Italy) and NextGenerationEU, which provided a Ph.D. scholarship to the lead author.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data analyzed are all publicly available and all the sources are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NLP	Natural Language Processing
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pretrained Transformers
NER	Named Entity Recognition
SQL	Structured Query Language
TTS	Text-to-Speech
MT	Machine Translation
UMT	Unsupervised Machine Translation

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3058.
2. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
3. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving language understanding by generative pre-training. *OpenAI* **2018**.
4. Chowdhary, K.; Chowdhary, K. Natural language processing. *Fundam. Artif. Intell.* **2020**, *1*, 603–649.
5. Otter, D.W.; Medina, J.R.; Kalita, J.K. A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 604–624. [CrossRef]
6. Qiu, X.; Sun, T.; Xu, Y.; Shao, Y.; Dai, N.; Huang, X. Pre-trained models for natural language processing: A survey. *Sci. China Technol. Sci.* **2020**, *63*, 1872–1897. [CrossRef]
7. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event, 3–10 March 2021; pp. 610–623.
8. Dang, N.C.; Moreno-García, M.N.; De la Prieta, F. Sentiment analysis based on deep learning: A comparative study. *Electronics* **2020**, *9*, 483. [CrossRef]
9. Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; Sen, P. A survey of the state of explainable AI for natural language processing. *arXiv* **2020**, arXiv:2010.0071.
10. Alyafeai, Z.; AlShaibani, M.S.; Ahmad, I. A survey on transfer learning in natural language processing. *arXiv* **2020**, arXiv:2007.04239.
11. Wu, L.; Chen, Y.; Shen, K.; Guo, X.; Gao, H.; Li, S.; Pei, J.; Long, B. Graph neural networks for natural language processing: A survey. *Found. Trends® Mach. Learn.* **2023**, *16*, 119–328. [CrossRef]
12. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
13. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
14. Shin, T.; Razeghi, Y.; Logan IV, R.L.; Wallace, E.; Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv* **2020**, arXiv:2010.15980.
15. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv* **2019**, arXiv:1901.02860.
16. Krause, B.; Kahembwe, E.; Murray, I.; Renals, S. Dynamic evaluation of transformer language models. *arXiv* **2019**, arXiv:1904.08378.
17. Ziegler, D.M.; Stiennon, N.; Wu, J.; Brown, T.B.; Radford, A.; Amodei, D.; Christiano, P.; Irving, G. Fine-tuning language models from human preferences. *arXiv* **2019**, arXiv:1909.08593.
18. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The efficient transformer. *arXiv* **2020**, arXiv:2001.04451.

19. Zhu, C.; Zeng, M.; Huang, X. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv* **2018**, arXiv:1812.03593.
20. Garg, S.; Vu, T.; Moschitti, A. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 7780–7788.
21. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 3088.
22. Wu, S.; Cotterell, R.; Hulden, M. Applying the transformer to character-level transduction. *arXiv* **2020**, arXiv:2005.10213.
23. Zhu, J.; Xia, Y.; Wu, L.; He, D.; Qin, T.; Zhou, W.; Li, H.; Liu, T.Y. Incorporating bert into neural machine translation. *arXiv* **2020**, arXiv:2002.06823.
24. Yasunaga, M.; Leskovec, J.; Liang, P. Linkbert: Pretraining language models with document links. *arXiv* **2022**, arXiv:2203.15827.
25. Hosseini, P.; Broniatowski, D.A.; Diab, M. Knowledge-augmented language models for cause-effect relation classification. In Proceedings of the First Workshop on Commonsense Representation and Reasoning (CSRR 2022), Dublin, UK, 27 May 2022; pp. 43–48.
26. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
27. Liu, Q.; Chen, Y.; Chen, B.; Lou, J.G.; Chen, Z.; Zhou, B.; Zhang, D. You impress me: Dialogue generation via mutual persona perception. *arXiv* **2020**, arXiv:2004.05388.
28. Guo, T.; Gao, H. Content enhanced bert-based text-to-sql generation. *arXiv* **2019**, arXiv:1910.07179.
29. Wang, Y.; Wang, W.; Joty, S.; Hoi, S.C. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv* **2021**, arXiv:2109.00859.
30. Clive, J.; Cao, K.; Rei, M. Control prefixes for parameter-efficient text generation. In Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), Online, 7–11 December 2022; pp. 363–382.
31. Xiong, W.; Gupta, A.; Toshniwal, S.; Mehdad, Y.; Yih, W.T. Adapting Pretrained Text-to-Text Models for Long Text Sequences. *arXiv***2022**, arXiv:2209.10052.
32. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The long-document transformer. *arXiv* **2020**, arXiv:2004.05150.
33. Liu, Y.; Gu, J.; Goyal, N.; Li, X.; Edunov, S.; Ghazvininejad, M.; Lewis, M.; Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 726–742. [CrossRef]
34. Xiao, W.; Beltagy, I.; Carenini, G.; Cohan, A. Primer: Pyramid-based masked sentence pre-training for multi-document summarization. *arXiv* **2021**, arXiv:2110.08499.
35. Baumel, T.; Eyal, M.; Elhadad, M. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv* **2018**, arXiv:1801.07704.
36. Ghalandari, D.G.; Hokamp, C.; Ifrim, G. Efficient Unsupervised Sentence Compression by Fine-tuning Transformers with Reinforcement Learning. *arXiv* **2022**, arXiv:2205.08221.
37. Wang, X.; Jiang, Y.; Bach, N.; Wang, T.; Huang, Z.; Huang, F.; Tu, K. Automated concatenation of embeddings for structured prediction. *arXiv* **2020**, arXiv:2010.05006 .
38. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
39. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 24–28 June 2022; pp. 10684–10695.
40. Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv* **2022**, arXiv:2204.06125.
41. Wang, C.; Chen, S.; Wu, Y.; Zhang, Z.; Zhou, L.; Liu, S.; Chen, Z.; Liu, Y.; Wang, H.; Li, J.; et al. Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers. *arXiv* **2023**, arXiv:2301.02111.
42. Li, C.; Xu, H.; Tian, J.; Wang, W.; Yan, M.; Bi, B.; Ye, J.; Chen, H.; Xu, G.; Cao, Z.; et al. mPLUG: Effective and Efficient Vision-Language Learning by Cross-modal Skip-connections. *arXiv* **2022**, arXiv:2205.12005.
43. Plepi, J.; Kacupaj, E.; Singh, K.; Thakkar, H.; Lehmann, J. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. In Proceedings of the The Semantic Web: 18th International Conference, ESWC 2021, Online, 6–10 June, 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 356–371.
44. Oguz, B.; Chen, X.; Karpukhin, V.; Peshterliev, S.; Okhonko, D.; Schlichtkrull, M.; Gupta, S.; Mehdad, Y.; Yih, S. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv* **2020**, arXiv:2012.14610.
45. Wang, W.; Bao, H.; Dong, L.; Bjorck, J.; Peng, Z.; Liu, Q.; Aggarwal, K.; Mohammed, O.K.; Singhal, S.; Som, S.; et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv* **2022**, arXiv:2208.10442.