УДК 811.93
DOI: 10.22213/2618-9763-2021-4-109-121

Sándor Ágoston Pál, Student
Kalashnikov Izhevsk State Technical University, Izhevsk, Russia
The Faculty of Mechanical Engineering, Informatics and Electrical engineering, Széchenyi István University in Gyor, Gyor, Hungary

"It is impossible to reject, to deny progress, this has never happened and never will be, it is irreversible. You need to learn how to manage artificial intelligence."
Vladimir Vladimirovich Putin

AI METHODS THAT CAN GENERATE LONGER TEXTS WITH MINIMAL INITIAL INPUT

The article is devoted to the description of the work and the research of text generators generate longer texts with a minimum input of primary information. The research is interdisciplinary of NLP and carried out at the intersection of Natural Language Processing and Computational Linguistics. The emergence of the growth of web-based textual information has significantly accelerated the development of some scientific fields that have existed for many decades, but in the absence of much access to input data, they have not developed as rapidly as in the past two decades. There have been several promising ideas and experiments in this field for the automatic processing of natural language texts, which have already been implemented in many of the systems used, including many commercial systems. In my research, I try to better understand and compare existing CL applications and their operation in each topic. However, specific applications fall into one topic, with little or no difference in their area of operation and application. The methods NLP in the field of artificial intelligence were described. The most important and traceable form of communication is writing. Artificial intelligence and automation are at the heart of the ongoing fourth industrial revolution. During the research process, the author drew several conclusions during the research process. Among other things, he understood the importance of data vectorization and that any abstract process can be well modelled with mathematical processes. Although the use of different text generators are based on AI, the role of man is not negligible, as setting input parameters and checking output results for different methods still requires human control to this day. The layout and "quality assurance" of the text is the responsibility of AI researchers. Various techniques and methods have evolved and supplemented significantly over the past 20 years, with the latest text generators almost surpassing the framework and quality of text written by a person. This does not mean that in the future we will only read machine-generated text. Intuition and creativity still require the presence of man to this day, the machine processes the set of information we enter or enter and processes the various patterns and then generates texts using them.

Keywords: natural language processing; artificial intelligence; fourth industrial revolution; generative pre-trained transformer 2; stochastic models.

Introduction

One of the basic tools of culture is writing and reading, which we create with the help of different signs. There are many languages in the world, so different punctuation (Cyrillic, Hebrew, Latin, Greek, Chinese, etc.) and word processing modes have developed. All of these have contributed to the development of the text-making culture that exists today.

However, the range of communication tools has undergone a major transformation in the last 50 years, which has resulted in the emergence of various digital info communication tools such as e-mail, SMS, and chat. Their user background is growing day by day and contributing to continuous improvement.

In today's increasingly digital world, we are confronted with a significant amount of data, day-to-day, through which, or for whatever reason, we try to shorten, or at least speed up, our own data generation. The communication path on the various electronic devices has expanded significantly, so we spend a lot of time on chat programs and various "social network" applications. Writing posts, posts and reactions and answers is primary in addition to keeping in touch, which we do directly through chat. Of course, I mean a general phenomenon here, not a statement of fact that is true for everyone. Trends, on the other hand, show that communication has shifted towards conversations over the Internet.

The exchange of text messages takes place on different platforms, and the wide range of applications used for this is expanding day by day. And programmers try to provide trend-appropriate ideas, solutions, and tools to choose their product. One way to enhance your user experience is to use a variety of auto-spelling and text-filling applications or auxiliary algorithms in different interfaces. The other larger group is provided by various translation software (international communication in languages we do not know) and graphical symbols (emoticons) and solutions.

Text generation, on the other hand, is not essential in these cases, so it is important for users to describe and predict the appropriate words in advance. However, in some areas, the generation of longer texts is of great importance, as it does not matter when and what we describe. These are various chat bots to make it easier for consumers to choose, automatically giving users the right answers. This is not only negligible in financial terms but also in terms of customer satisfaction. Just think that a 0-24-hour software is constantly available to consumers while accurately informing the consumer about the service or product they are looking for.

Today's modern and embarrassingly well-functioning text generation required several leaps and bounds. To this end, advanced artificial intelligence-based solutions have been developed, and these software/algorithms are already at a level that has been able to pass the Turing test. The reason for this is that a person is unable to tell from the text generated by MI that it was written by a person or machine.

The emergence of the growth of web-based textual information has significantly accelerated the development of some scientific fields that have existed for many decades, but in the absence of large amounts of input data, they have not evolved as rapidly as in the past two decades. Automatic word processing and computational linguistics are my two main subjects. There have been a number of promising ideas and experiments in this field for the automatic processing of natural language texts, which have already been implemented in many of the systems used, including many commercial systems. The scientific results of the field can also be found on the website of the International Association of Computational Linguistics, which summarizes the work of several scientific conferences in this field from 2002 to the present.

Computational linguistics is an interdisciplinary field that has emerged at the crossroads of sciences such as linguistics, mathematics, computer science, and artificial intelligence. Computational linguistics is closely related to the field of artificial intelligence, within which software models for individual intellectual functions are developed.

Somewhat simplified, the task of computational linguistics can be formulated by developing methods and tools to construct language processing for various applied problems for the automatic processing of texts using artificial intelligence. The development of a language processor for an applied problem includes an official description of the linguistic properties of the processed text (at least the simplest) that can be considered a text model (or language model).

The boundaries of artificial intelligence are not yet known, as we cannot delimit them properly. In addition to learning algorithms, there may come a time when different algorithms and systems will be able to learn independently and retrieve information and databases independently, thus being able to run as autonomous programs free from human intervention. And these types of algorithms can usher in an era of text generation-interpretation-creation in which they can fully satisfy different user needs without keywords.

Overview

The basis of text generation is the transformation of structured data into a specific natural language (so the aspect of text translation is also important).This can be the generation of additional text from short sentences, a summary of a particular text, but also question-and-answer interactions during interactive conversations (such as a chatbot).

One of the major milestones in text generation is the emergence of mobile phones, as the first devices launched were not very large, so writing SMSs was quite cumbersome. Not to mention that it wasn't suitable for a full QWERTY keyboard layout, so more letters were added to one button (3*4 matrix keyboard solution).This process is called multi-tap, which, as its name implies, results in a letter or character being issued as a result of multiple prints [1].The problem started there, in countless cases, you had to press the keys more than once to retrieve a character, which means a lot of "redundant" impressions. Thus, this process has made text entry infinitely slow and difficult to edit, now not addressing young Asian people with rifles who have pocketed prizes from mobile word processing in various cash prize competitions [2].

Thus, writing SMS has a word-to-minute speed that can be significantly increased by using a text generator, also called predictive text input or T9 (Text on 9 keys), these were the first commonly used text generators, although it is better to use the term generator in this. case. This procedure, on the other hand, had a weakness, and a combination of letters yielded several words, but they appeared in proportion to the frequency. That is, we could not always get the word we wanted to describe while writing, that is, the prediction does not depend on the context [3].

To solve the problem, a number of newer and additional software were used, which were already able to complete the word or complete the word, so these solutions could be much more accurate. Interestingly, these solutions are still present to this day, as the words you type in browsers and search engines are automatically supplemented and listed by these extensions [4].

A language model is basically a machine learning model that can analyse a part of a sentence and predict the next word from it. The most well-known language models used in everyday life are the built-in recommendation features of smartphone keyboards, which suggest the next word based on the text you just typed.

Probability models have several advantages. They can be conveniently taught based on data: learning consists only in counting occurrences (with some tolerance due to errors caused by small sample size).Furthermore, they are more robust (since they accept any string of characters, albeit with a low probability), reflecting the fact that not 100% of speakers agree on which sentences are part of the language; and are suitable for resolving ambiguity: the most probable interpretation can be selected based on probability. The probabilistic language model defines a probability distribution over a (possibly infinite) set of strings. Probabilistic language models are clearly structured on a statistical basis. Of course, there are several statistical methods within this [5].

More advanced predictive solutions already allow automatic word saving, a kind of “dictionary” one of the best solutions provided by Adaptxt. Its greatest advantage is that it is able to recognize, save and recall not only words but also complete sentences [6]. An added benefit is that it not only checks the text we write, but also the text we receive (SMS, email, chat, etc.) and saves new words and frequently occurring sentences.

Adaptxt already “estimates” on the basis of context, so it determines the possible words and sentence parts from the sentence structure. These software uses dictionary-based solutions and work with rudimentary algorithms (compared to today’s AI solutions) [7].

Among the non-dictionary-based solutions, an application called Letter Wise stands out, which is based on calculating the probability of occurrence of letters, ie it is not based on a dictionary, but on an extensive probability calculation database. It estimates words based on prefixes, ie when you press a certain letter, it offers the most probable next letter. If the offer is not suitable for us, it puts the second, third or fourth most probable letter in front of the line, which significantly increases the speed of text generation [8].

The interplay between AI and linguistics has led to different disciplines and trends, including computational linguistics [9] and natural language processing (NLP) [10]. Thus, at the crossroads of two very complex disciplines, different AI models have emerged that have developed advanced linguistic comprehension that can successfully interpret not only sentence structures but also linguistic context and topic. Originally, speech recognition and language comprehension were the basic problems to be solved, since then natural language / text generation has also been included in research directions [11].

The first solutions for language processing were for the most part the statistical methods/algorithms on which the methods and software presented above were based. These were created using a large number of different text corpora. Machine algorithms most often generated decision tree analysis that linked additional character (s) to the input data based on probability. One of the models of algorithms based on statistical methods is the Markov model, the application of this method has greatly facilitated the generation of longer sentences [12].

The significance and advantage of these models is based on their robustness, as they are able to produce “good” results even with erroneous input data, i.e., the first generations of chat bots were born with such solutions [13].

Chat bots are basically applications that are able to have a meaningful conversation with the user. Of course, there is a wide range of these apps, as there are AI-based chat bots for writing stories from simple answers to emotionally moving stories. We do not know the exact date on which chat bots will be released, but we unanimously consider the beginning of the 2010s to be one of the specific forms of artificial intelligence used in business. The proliferation of chat bots has been greatly facilitated by the daily use of online chat programs, so chat bots running in chat programs no longer appear [14].

The spread of neural networks, on the other hand, has been greatly aided by the fact that there is no need for complicated parameter design, i.e., the transition from statistically based text generation to neural network-based text generation. The use of neural networks can be seen as a major paradigm shift, as through deep learning, AI systems automatically learn the relationships between parts of a text without the need to model them [15].

With the help of these new methods, the generation of natural, readable language, first-rate AI language proficiency, and answering questions have now been solved to some degree, and even the generation of a book has become feasible, although semantic problems still limit the standard [16].

Presentation of traditional/old-fashioned solutions

Markov chain-based text generators

Before I introduce the concept of Markov chains, I will briefly recall the basic concepts and axioms of probability theory. Primarily in the field of mathematics, the random variable *X* is a quantity that results in one or more random phenomena or events. The events of these phenomena can be the result of a number (or "numeric", such as vectors) or just what I correspond to. For example, we could define a coin toss as a random variable, where 0 is the head and 1 is the write (or vice versa).I would also mention, for example, that the outcome of the possible results of a random variable can be discrete or continuous: for example, a normally distributed random variable can be continuous, and the Poisson random variable distribution can only be discrete.

We can define a stochastic process as a set of indexed random variables within a set T that often denotes different moments of time (I will refer to this in the following).In the two most common cases: T can be either a set of natural numbers (a random process with discrete time) or a set of real numbers (a random process with continuous time).For example, if I toss a coin every day, I define a random process with a discrete time, while if the continuously changing value of a stock on the stock market determines a random process, it happens with a continuous time. Random variables at different points in time may be independent of each other or have some dependence; in addition, they may have a continuous or discrete state space.

Stochastic processes can be divided into well-separable groups, for example:

– Gaussian processes
– Poisson processes
– Autoregressive models
– Moving average models
– Markov chains.

The basis of Markov chain models is thus described by a stochastic model that gives the possible outcomes/probabilities of a given event chain (series) in the light of the event before it. Markov chains, models where a change between certain s states can occur with probability p (i), where i denotes a change between two s states. A state can even maintain its own state with a certain probability, it does not have to change. The sum of the probabilities of a state change starting from a status is always 1 (100 %) [17].

This model can be used in several predictable tasks, including Bayesian statistics, thermodynamics, statistical mechanics, information theory, and artificial intelligence research and applications. This model is used in several places in linguistics and artificial intelligence applications, such as speech recognition. Markov models are also used for text generation, which can be used to create superficial but realistic texts, but in this case a sample document is needed to generate a new one by analysing the pattern. Many Markov chain-based open-source text generation databases are available on the Internet, including the so-called RiTa Toolkit.

Each of these individual cases has certain well-defined properties that allow us to better analyse and understand. One property that greatly simplifies the study of a random process is the "Markov property".

Informally put, the Markov property states that if we know the result achieved by a random process at a given time, we can get more information about the future behaviour of the process, we don't have to collect other information about the past outcomes of the process. More precisely, at any point in time, the conditional distribution of future states in a process depends only on the current state, not on past states. A random process with Markov properties is called a Markov process.

Mathematically, the Markov model can be represented as follows:

$$X = (X_n)n \in \mathbb{N} = (X_0, X_1, X_2, \ldots) .$$

Where the process takes its values from a discrete set E at any point in time:

$$X_n \in E \qquad \forall \in \mathbb{N} .$$

Based on these, the meaning of the Markov chain is:

$$P\left(X_{n+1} = s_{n+1} \middle| X_n = s_n, X_{n-1} = s_{n-1}, X_{n-2} = s_{n-2}, \ldots\right) = \\ = P\left(X_{n+1} = s_{n+1} \middle| X_n = s_n\right)$$

.

The last formula reflects the fact that at the timeline (where I am now and where I have been before), the probability distribution of the next state (where I will be at the next time) depends on the current state, but not on past states [18].

Outputs and input parameters

To generate the text, we first need texts that will allow our Markov chain-based algorithm to work. To do this, we need a syntactic n-gram that does not depend on the linear structure of the text, i.e., it is able to form infinite n-number dependencies on different parts (phonemes, letters, words, or word combinations) in a decision tree model based on a given text [19].

Take a biblical passage as an example:

"*For God so loved the world, that he gave his only begotten Son*" (John 3:16).

Let our n-gram be n = 3, in which case we generate 3-gram (trigram) passages based on the words, which would look something like this:

For God Loved→God so loved→loved God so→etc.

If the letters are taken as a basis, up to 3 letters of "word fragments" can be formed, for example, "*loved*" *Ved →ove →oed→lde*etc...

A text generator must generate "quasi" meaningful text, as this is the only way to meet our requirements. To do this, we need a method that, after the first random word in the sentences, the algorithm puts the right words (letters) and closes the given sentence at the right point. The text we enter, in this case the biblical sentence, along with the n-grams, will be able to be recognized by the algorithm based on the source and generated from them.The algorithm of the Markov chain model will be able to select the appropriate word from the generated word collection after the random start word, but this will only happen on a probabilistic basis. The resulting text is mostly pseudo meaningful but is not guaranteed to say anything to the average reader.

I will present the model through a simple example. I'm looking at the daily traffic of a fictional library. There are 3 possible states every day:

– The reader will not visit the library that day (*N*);

– The reader visits the library but does not read a book (*V*);

– The reader visits the library and reads an entire book (*R*).

So, we have the following state space:

$$E = \{N, V, R\}.$$

Suppose on the first day, this reader has a 50% chance of just going to the library, and a 50% chance of going into the library and reading at least one book. The vector describing the original probability distribution ($n = 0$) then looks like this:

$$q_0 = (0.00, 0.50, 0.50).$$

The following probabilities can also be observed:

When a reader doesn't go to a library one day, they're 25% likely not to go the next day, 50% likely to just go there, and 25% more likely to go and read a book (books).

When a reader goes to a library one day but does not read, there is a 50% probability that he or she will go to the library again the next day and not read a book, and a 50% probability that he or she will go and read a book (books).

When a reader goes to the library one day and reads a book, there is a 33% probability that he or she will not go the next day, a 33% probability only goes to the library, and a 34% probability of visiting the library and reading the book.

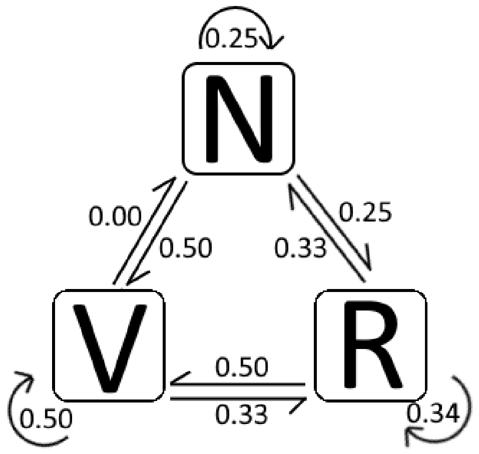
The following transition matrix is then generated:

$$p = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ 0.00 & 0.50 & 0.50 \\ 0.33 & 0.33 & 0.34 \end{bmatrix}.$$

Based on the above, you can know how to calculate the probability of each condition for this reader on the next day ($n = 1$)

$$q_1 = q_0 p = (0.00, 0.50, 0.50) = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ 0.00 & 0.50 & 0.50 \\ 0.33 & 0.33 & 0.34 \end{bmatrix} = \\ = (0.165, 0.415, 0.420).$$

The probability dynamics of the Markov chain for the event given in the previous example can also be plotted as follows (picture):



Example task (used program: PaintDotNet)

Limitations of the method

Random processes are collections of random variables that are often indexed over time in the case of a random process, the Markov property means that the probability of a given event occurring in the future does not depend on the past. The discrete-time Markov chain is a random process with discrete time indices that satisfy the Markov property. The Markov property of Markov chains greatly facilitates the interpretation and analysis of these processes, and allows the derivation of various interesting results (mean return time, stationary distribution...).

The Markov chain proves to be very effective for statistical predictions and problem solving.

The sentences generated in this way can be mostly meaningful in themselves, but there will be no proper semantic cohesion between the sentences. The word turns and connections in the original text will appear in the generated text. The other problem is caused by the fact that in many cases repeated word connections will develop, that is, it will appear more than necessary in a given sentence. This may be because given words appear more than once in a given text (now moving away from the example) will significantly increase the probability in this Markov chain model, so we will get the sentences in the output with a statistical probability rather than a grammatical probability.

To reduce the problems, you can choose from several options, one of which is to feed longer text, as n-gram can generate more word turns. For generation, it is important to specify a low n value so that different parts of the original text will be less repetitive. The biggest disadvantage of statistical text generation lies in its advantage. Although it is able to assess the input text on a stochastic basis and give personalized word phrases as output, it is unable to understand the context and underlying saying.

The widespread and widespread use of artificial intelligence has opened up new ground for NLP. Developers don't have to rely on old stochastic models that were often only able to create seemingly meaningful texts. Stretching the boundaries of the model is already exhausted in text generation, as it is not specifically designed for this. The Markov model is still outstanding in statistical fields, but it is only tangentially useful in text generation. Any newer model of artificial intelligence can perform any task of the model excellently and even better.

SWIFT-Key machine-learning algorithm for a virtual keyboard

Overview of the method

The cell phone is almost out of touch with people, we write more on our cell phone than we actually call. You don't usually have to press the screen or keyboard to type, just slide your finger. The best solution for this input method, supported by artificial intelligence, is the Swift Key keyboard application. Although it is a fact that they had ancestors like Swype, these applications are not as advanced as Swift Key, which uses the fully developed Neural Alpha [20].

Microsoft Swift Key is a virtual keyboard application originally developed by Touch Type for Android and iOS devices. It first appeared on Android in July 2010, and then an iOS release was introduced in September 2014.The greatest virtue of Swift Key is the machine learning algorithm. We look at it as part of AI, the point is that the constructed algorithm can automatically improve using experience and data. Thus, based on user habits, machine learning algorithms make predictions or decisions based on textual data entered by the user without being specifically programmed to do so [21].

These ML algorithms rely on statistical models, so in order to apply them, we must first write our own model adapted to the given task. After that, a specific textual basis must be given, called training (beginner) data, without which the created algorithm will not be able to perform its main task. However, after the "learning" period, it will draw appropriate conclusions from "unknown" data [22].

The types of machine learning can be divided into 3 categories, supervised, unsupervised, and confirmatory learning. SwiftKey moves between the first category and the second category, as the system first receives the input and output data "tagged" and first the software developer feeds a dictionary and regulates the output data to be displayed with grammatical knowledge specific to the given language (so to speak). Swift Key will have language skills). Then the user starts using his phone, including chatting, browsing, and other activities that provide a kind of learning period for Seift Key [23].

This is followed by a period of unsupervised learning during use, as in more than one case Swift Key adds different additions to a given sentence during certain activities. That is, when a new activity is started by the user, the algorithm "in" inherits and completes the text.For example, a user uses Facebook and Whatsapp first, then after years starts reading and searching for RSS feeds and emailing, and the keyboard uses the same word phrases as when writing an official letter.

This is, of course, due to the fact that SWIFT Key is present as a cloud-based service (Swift Key Cloud), so by learning and interpreting different users and usage styles, it can provide useful input data to our phone from other input data. The seventh improved version of the Swift Key software and its algorithm already exists, in which the gesture control and a new toolbar have already been introduced [24].

Limitations of using Swift Key

Unfortunately, there are no resources available to me about the SWIFT Key architecture, so I worked from existing sources. The software has a basic vocabulary that can be continuously expanded with different variations of words. This vocabulary is currently available in 29 languages in the latest version. In addition, it has a machine learning algorithm that analyses the relationships between input characters and parts of text and performs probability calculations. This system is connected to the Swift Cloud, which can retrieve words or parts of sentences based on relations using a kind of database manager [25].

Machine learning algorithms have brought a breakthrough in personal text generation. In some cases, however, they are unable to articulate the expected text. Which factors may be behind the algorithm, i.e., the algorithm does not get the right amount and quality of data, for example we do not use the Swift Key keyboard enough. Data is distorted, for example, young people also display the word phrases used within their generation – through the cloud – on older devices. Data security issues and incorrect evaluation of the evaluation function of the algorithm may occur [26].

Although Microsoft claims the data described with Swift Key is absolutely secure, the problem for users is that cloud-based storage of your data can go anywhere. Just think about it, the model develops based on the labelling and statistical analysis of the entered data, so this data is also included in the development data set. Cloud-based storage has evolved a lot, but it still carries some dangers.

The basic goal of Swift Key was to generate complete, meaningful sentences from a few initial inputs, but the project took a completely different direction. The latest version of Swift Key has already combined the statistical model and the Neural Alpha labelling method, making predictions for users much more personalized and more contextualized through artificial intelligence. Using Neural Alpha, Swift Key tries to understand the context of the text you type and offers prediction for it.Neural Alpha conducts machine learning through a neural network and thus analyses input language elements similarly to the functioning of the human brain. The real goal, which is to generate longer and more meaningful texts, is getting closer, but the day has not yet come.

Presentation of modern solutions

GPT-2 Algorithms in Open AI Development

Purpose of algorithms

Open AI has developed two AI systems that generate text for a complete dissertation from a single sentence.These algorithms focus on language modelling, so always try to guess what word will follow that word. As the most robust text generator of today, these two systems are considered to be capable of generating text in many styles and forms [27].

Based on the above, we could also conclude that the GPT-2 is basically the next word prediction function of the telephone keypad application, but in fact the algorithm that is the subject of the topic is much more robust and sophisticated.GPT-2 is built on a huge data set called WebText, which was created by OpenAI researchers based on Internet research. As a benchmark, the Swift Key takes up 78 MB of space, the smallest version of the trained GPT-2 requires 500 MB of storage space to store all parameters, and the largest version of the GPT-2 requires 13 times more space, around 6.5 GB.

It's no coincidence that GPT-2 quickly became one of the most popular successes in deep learning. Some have written books and poems, while others have used GPT-2 to create video games, such as AI Dungeon 2, an almost infinitely versatile text-based role-playing game that garnered $16,000 a month on Patreon.

He output and input parameters

Built on the foundations of its GPT-2 predecessor, the key difference is that it has been trained by the makers to generate words one after the other from 40GB of internet data sources. The exact structure and beta of the algorithm is not public because the developers say the risk of malicious use is too high.

The smaller framework software, with approximately 1.5 billion parameters, was trained, and its quality was also highlighted. The input data set focused on the widest possible diversity. This was achieved through the use of pre-edited and filtered websites, more specifically Reddit's 5/3 rated articles, which were considered by developers to be the quality we would expect for demanding and eye-catching texts. The GPT-2 algorithm is designed to "predict" the word after the word you type, but at the same time take into account the words described earlier, so it can also observe the context. This can be considered as a kind of semantic problem solving, as the wide and varied data set provides an opportunity to generate longer and more meaningful texts [28].

The GPT-2 is also equipped with a learning algorithm that allows it to learn from other text generating applications, thus using a larger amount of text without having to learn it beforehand. In addition, details of the algorithm used to solve language problems that can summarize texts and answer questions have been added to the system, although their complexity has not yet been investigated. The learning process is unattended, so there is no need to label individual input data, which gives the system more freedom and efficiency [29].

The GTP-2 system artificially generates text patterns in a variety of styles and then displays the appropriate text in the output results. This allows the algorithm to generate different coherent texts in a given style from a given topic. The adequacy of the input pattern can be improved by pre-teaching the algorithm, i.e., the production of specific topics can be expected with the knowledge of appropriate similar patterns. Another solution is to improve the recognition of the algorithm after repeated attempts and learning, but the human control also helps the operation of the algorithm, as it is possible to load data sets or specify other parameters related to the solution of the given problem.

However, during testing, as with any man-made system, errors can occur. It also formulates primarily repetitive text and illogical comparisons (e.g., underwater fire), in addition, unwarranted changes of subject appear as typical errors in the generated texts. For language tasks (such as answering questions, reading comprehension, summarizing, and translating), the system processed the data in outstanding quality. This was achieved by coordinating the trained models and the different algorithms, but the results were unsatisfying compared to the algorithms specifically trained for this purpose [30].

Operation of main components

For GTP-2 based text generation, you need to select a dataset that is also provided by OpenAI (WebText), here it is worth using a text database that is relevant to the current text generation. Next, we need to create a so-called keyword list, for which there are several algorithms available, but the simplest is a TFIDF[1].

This is followed by so-called fine-tuning, where the various software tools and algorithms need to be concatenated into one system, one of the essential steps of which is tokenization, which is mandatory in the handling or generation of textual data. Skipping this step will make processes with text data impossible. Tokenization is one of the general tasks of NLP, which is to separate a given text into chunks of text (tokens).Tokens can be characters, words, and the n-gram characters shown earlier. Thus, the corpus can be made usable during tokenization [31].

Thus, during the fine-tuning, after tokenization from the data set, the data sets and other parameters to be used are determined, and then the keyword list is classified on the text corpus and text generation is possible. The texts that are generated in this way are varied and vary depending on the size of the data set, although their coherence is still questionable.

Neural network solutions in text generation – BERT model

Description of the model

The BERT model, currently considered by researchers to be the most advanced model, is the so-called “state-of-the-art” method, along with the most researched. This AI system was created by Google. This method of text generation, extracting or summarizing, is very important, as in many cases an accurate and quick summary of a large amount of text is of paramount importance, for example, in making a medical diagnosis or preparing an economic decision.

BERT is a neural network from Google that can provide a solution to a wide range of researched problems. BERT can be used to create artificial intelligence programs to process natural language. These programs can answer free-form questions, create chat bots, machine translators, analyse text, and more.

The essence of the summary can be approached from two different sides, one possibility is to extract an extract from the original text using an abbreviated but identical vocabulary, this is called an abstract approach. There are two pillars to compiling this type of summary with AI, as the summary should not only meet the grammar criteria, but also include the actual content of the original text. Another way to summarize using the AI system is to extract the “sentences” from the original text to extract the individual sentences that best express the content of the original report. This solution is much easier, as in this case it is not necessary to deal separately with issues of grammar.

The BERT model stands for “Bidirectional Encoder Representations from Transformers”, which covers a pre-taught language model with different word, sentence, and position recognition and usage based on a text corpus derived from a very large (800 million word) database. Originally, this model and similar systems were developed to process language problems such as sentence or paragraph interpretation and translation. BERT is the most efficient NLP model besides the GPT model.

Google Research released BERT's open-source implementations on Tensor Flow and provided the following pre-trained models[32]:

BERT-Large, Uncased (Whole Word Masking): 24-layer, 1024-hidden, 16-heads, 340M parameters

[1] A statistical tool for quantifying short keywords, weighting terms in a given corpus of text.

BERT-Large, Cased (Whole Word Masking): 24-layer, 1024-hidden, 16-heads, 340M parameters

BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters

BERT-Large, Uncased: 24-layer, 1024-hidden, 16-heads, 340M parameters

BERT-Base, Cased: 12-layer, 768-hidden, 12-heads, 110M parameters

BERT-Large, Cased: 24-layer, 1024-hidden, 16-heads, 340M parameters

BERT-Base, Multilingual Cased 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters

BERT-Base, Multilingual Uncased 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters

BERT-Base, Chinese: Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

These versions are all freely available and can be experimented with in a simple package in PyTorch.

Output and input parameters, processes

The first step is to teach the BERT model as a deep learning algorithm, this training is two-step as in the GTP-2 system mentioned earlier. The first step is "pre-training", in the process of which we create a general representation based on the input data and perform the fine tuning as the second step based on the output parameters. The simplest way to formulate this is to develop a problem-specific model and thus optimize the accuracy and semantic fineness of text generation.

A given corpus can be used for both teaching and testing, these parameters can be adapted to the current problem. The usability of the BERT model is suitable for both summary generation and new text generation. In the extractive solution, in the BERT model, the algorithm must be fine-tuned to rank the sentences for the given title and context, thus selecting the most appropriate sentences to summarize.

If you want to send text to the input of a neural network as input, it must be represented as numbers. The easiest way to do this is to feed it into the neural network. Each letter is then coded with a number between 0 and 32.

You can get much better results by not submitting one letter at a time, but an entire word or syllable. So, we can't make the mistake that Letter Wise made (instead of predicting the word prediction).The simplest option is to compile a dictionary of all the existing words, feeding these words into the neural network. For example, if the word "telephone" is in position 666 in this dictionary, the number 666 is entered into the neural network input. When we talk orally and use the word "phone", a lot of associations appear in people: "smart", "iPhone", "Android". The BERT model provided an opportunity for associations to be coded as well. Simplified, the solution is that the words that are close to each other are placed next to each other in the database. In practice, each word is assigned several numbers, such as a vector of 64 numbers. And we will be able to measure distances as the distances of the points that these vectors point to in their respective dimensional space.

This method allows you to compare a word to multiple words or words that are close to meaning (depending on which axis you are examining). Moreover, we can perform arithmetic operations on words with vectors. Classic example: if you subtract the "man" vector from the "king" word vector and add the "woman" word vector, you get a certain vector result. Interestingly but not surprisingly, the word belonging to the resulting vector will match the Queen. Words can be represented by vectors in several ways, these methods have gradually evolved: word2vec, GloVe, Elmo. The representation and processing of words with vectors has already been used in the development of Swift Key.

We can see that the newer methods are all a huge improvement over their old ones, but they are still not able to work flawlessly in an autonomous way. This will of course change over time. Human intervention and task-specific alignment will be less and less necessary as, as we have seen, models learn and evolve during use.

Operation of main components

BERT consists of three basic components. First, it is from a pre-trained language model. Second, BERT has the decision-making power to decide which outcome is most likely. Transformer is a neural network architecture that allows the model to consider the entire context at once. The transformer displays the sentences in a more expressive way (tree-structured).Each layer of the neural network establishes many parallel connections between certain words, ignoring the others. The tree-like representation of sentences allows transformers to model contextual meaning and effectively study the relationships between distant words in complex sentences (this is possible through vectorization).

The third basic but most important component, unlike other language models, the BERT model reads both from right to left and from left to right at the same time and tries to predict which words are missing from the sentences (Masking).This bidirectionality encourages the neural network to process as much information as possible from any subset or subsets of words. Masking is not new in NLP, but BERT is the most effective in this triple blend (Prerequisite Model, Decision Ability, Bidirectional Reading Context Understanding).

The BERT-Base model uses a vocabulary of 30,522 words. The tokenization process here means that the input text must be divided into a list of tokens available in the dictionary. In order for the model to work with words that are not in the dictionary, BERT uses a WordPiece-based tokenization method (BPE).It gradually breaks a word that is not in the dictionary into smaller parts;that is, it represents a word in subsets. Since the parts of the word are already in the dictionary, we learn the context representation of the parts and the context of the word will be a combination of the contexts of the parts. This allows the model to process words that are not included in the training corpus.

The operation of the Bert model is transformer-based, covering an observation mechanism that allows it to recognize context based on words or word fragments. At idle, the transformer uses two different mechanisms. An encoder that reads text input and a decoder that makes a prediction. Guided models read the words from left to right or right to left sequentially, the BERT transformer encoder reads the entire sequence of words at once.The transformer is non-directional, but thus actually bidirectional. This capability of BERT allows it to understand the context of the text.

The input is a series of tokens that are first embedded in vectors and then processed by the neural network. The output is a series of vectors that get the indexes of the input tokens. Before teaching language models, you need to determine your future prediction goal, which is never an easy task. Many models predict the next word in a sequence as if you had to insert the word or words into a template. This results in a lack of understanding of the context and learning from the models [33].

Differences/similarities between older and new AI methods

Older methods operate entirely on a statistical basis or on a deep learning basis using probability calculations and decision tree structured algorithms. These solutions are still in use today, as newer techniques and solutions are much more resource intensive, although the error rate of the old methods is noticeably and numerically higher. The length and comprehensibility of the generated texts, although not in one case, is an unfavourable output for the users.

The methods used require very complex algorithms and statistical backgrounds, which can be downloaded as packets. For them to work properly, whether old or new, a large amount of text corpus or vocabulary is not enough, but a number of parameters and problem-specific settings are required.

In the case of the new solutions, the issue of tokenization and fine-tuning can also be said to be completely subjective, the end result producing a qualitatively better result. The authenticity of the solutions cannot be easily checked, because in more than one case it is possible to generate several synthetic texts with the same run parameter, i.e. we always get new text. In summary, whatever method we choose to provide the right background and problem-centric settings, the generated text can reach the level that is right for us.

This version of Swift Key was originally released in 2010.In 2016, however, a neural network was added to Swift Key, but the old statistical method was not discarded either. It can be seen and felt that this update has greatly increased the efficiency of the model. We can see that the old and new methods are very different but both are effective in their own way. The Swift Key update was a big step forward because it retained the benefits of statistical customization and neural network context recognition capabilities.

In my opinion, the right combination of old and new methods can achieve the highest degree of efficiency in text generation.

Results

Although the use of different text generators are based on AI, the role of man is not negligible, as setting input parameters and checking output results for different methods still requires human control to this day.The layout and “quality assurance” of the text is the responsibility of AI researchers.

Various techniques and methods have evolved and supplemented significantly over the past 20 years, with the latest text generators almost surpassing the framework and quality of text written by a person. This does not mean that in the future we will only read machine-generated text. Intuition and creativity still require the presence of man to this day, the machine processes the set of information we enter and processes the various patterns and then generates texts using them. In more than one case, it is not a man-made text.

The sophistication of text generation is induced by continuous problem solutions, as we have moved from simpler tasks (such as stacking characters one by one) to more complex solutions (the GTP-3-based AI system also writes blogs and program code). Different text generation is based on different user needs, which are increasingly moving towards mind-driven writing [34].

Future AI solutions will directly convert brainwaves into text, images, although this is still in the experimental phase. In my opinion, they will be solvable in 20-30 years. This solution can open up completely new directions for us. Look at the using a text generator to generate text based on the topic title [35]:

AI methods that can generate longer textswith minimal initial input (by Infer Kit)

"*AI methods that can generate longer texts with minimal initial input will be more effective than ones that can only generate short texts based on the length of the input text.Finally, the human reader is a constrained resource: the more text she has to read the less efficient her capacity to read will be, and she will require fewer sentences to read.Thus the quality of a given reader's engagement with the text will be a factor in determining the overall success of the AI method*".

The text generator used recognized the language correctly, but the interpretability of the text is highly questionable. So, it is clear from the text above that while there are some really compelling solutions, it is still a task for the near future to introduce their general use and exploit their usefulness.

It is my personal belief that there are two possible options for the future. Man and machine complement each other in the task solution or the machine performs tasks autonomously without any human intervention. I think there is a very good chance for the latter to operate autonomously in the field of text generation.

Conclusion

While writing the topic, I came up with a lot of things that weren't evident to me so far. The first time I dealt with Swift Key, I realized that even I could create a statistically based prediction algorithm. On second thought, this is not a short task, but with a good investment of time, it would be possible to create a completely usable and statistically teachable algorithm. I also really liked the way that text data must be vectorized socan be analysed more accurately, this is the text2vec method.I have understood that data, in whatever form, should be processed in the most efficient way in a mathematical way. My own idea came up when dialogues between students and teachers in the online space became significantly more difficult during the pandemic. I sensed that responses arrive very slowly due to processing time and interpretation.

I was wondering if an interface or application could be implemented that could interpret students 'questions based on a template with language processing. Here, the student could ask the instructor a question by filling out the template. The instructor would make a few template-based status reports each day if needed. Students would receive details from these status reports after processing or based on their questions. All that is needed for this system is for students to ask their questions exactly according to the given criteria and there should be no errors in the questions or the formula. Of course, this is very difficult to circumvent, so there should be a learning process that can filter out the most common errors in correspondence so that the system can handle a certain margin of error. If we wanted this system to work well, both students and faculty would have to handle the interface very regularly. If we could all meet the criteria, a lot of unnecessary misunderstandings and problems could be avoided from both sides. In fact, I believe that the complete replacement of the student information system with a single surface using artificial intelligence methods would be a huge step forward in many ways. It would be much more transparent, manageable and interactive.

Such a portal would be informative for students in both primary and higher education. Both students and teachers could benefit enormously from the large amount of data that is generated while using the portal. The portal would be able to inform students and teachers about learning/teaching methods, language processing could summarize topics, highlight them, or simply generate content. While I was working on GPT2, I also looked at GPT3.I have seen that GPT3 could easily perform most of the tasks mentioned above. As an example, text summarization and highlighting already works in a very sophisticated way here.

Creating such a portal would create equal opportunities for all in all areas of education. Artificial intelligence could also perform very basic teaching functions in primary schools. In a high school, it could be used to generate questions, exam queues, so there would be no identical exam queues, or the question would be worded differently, and the teacher would only have to specify the topics. I believe that these methods and solutions will soon replace people in a lot of areas. The man will only be needed here at the supervisory level.

References

1. Rubya S., Monir S., and Ferdous H. S. Genetic approach to a flexible cell phone keypad with reduced keystrokes and key jamming for better human technology interaction. J. Multimed, vol. 7, no. 5, pp. 341-352, Oct. 2012. DOI: 10.4304/jmm.7.5.341-352.

2. Sang-Hun C. Rule of Thumbs: Koreans Reign in the Texting World - The New York Times. The New

York Times, 2010. Available at: https://www.nytimes.com/2010/01/28/world/asia/28seoul.html?th&emc=th

3. James C. L. and Reischel K. M. Text input for mobile devices. Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '01, 2001, pp. 365-371. DOI: 10.1145/365024.365300

4. Sandnes F. E. Reflective Text Entry: A Simple Low Effort Predictive Input Method Based on Flexible Abbreviations. Procedia Comput. Sci. 2015, vol. 67, pp. 105-112. DOI: 10.1016/j.procs.2015.09.254.

5. Artificial Intelligence in a Modern Approach Second, Revised, Extended Edition - Russell, Stuart Nor-vig Peter, pp. 737-738.

6. Qafmolla N. Automatic Language Identification. Eur. J. Lang. Lit., Jan. 2017, vol. 7, no. 1, p. 140. DOI: 10.26417/ejls.v7i1. P. 140-150.

7. Dunlop M. D., Durga N., Motaparti S., De Meo R., and Dona P., Open Adaptxt: An Open Source Enabling Technology for High Quality Text Entry, in Chi 2012 Ea, 2012.

8. MacKenzie I. S., Kober H., Smith D., Jones T., and Skepner E., Letter Wise: Prefix-based disambiguation for mobile text input. UIST (User Interface Software. Proc. ACM Symp.). 2001, p. 111.

9. Radev D. R., Joseph M. T., Gibson B., and Muthukrishnan P. A bibliometric and network analysis of the field of computational linguistics. J. Assoc.Inf.Sci. Technol., Mar. 2016, vol. 67, no. 3, pp. 683-706. DOI: 10.1002/asi.23394.

10. Nadkarni P. M., Ohno-Machado L., and Chapman W. W. Natural language processing: An introduction. J. Am. Med. Informatics Assoc., Sep. 2011, vol. 18, no. 5, pp. 544-551. DOI: 10.1136/amiajnl-2011-000464.

11. Cambria E. and White B. Jumping NLP Curves: A Review of Natural Language Processing Research. IEEE Comput.Intell.Mag., May 2014, vol. 9, no. 2, pp. 48-57. DOI: 10.1109/MCI.2014.2307227.

12. Kang M., Ahn J., and Lee K. Opinion mining using ensemble text hidden Markov models for text classification. Expert Syst. Appl., Mar. 2018, vol. 94, pp. 218-227. DOI: 10.1016/j.eswa.2017.07.019.

13. Steven A. Data-Intensive Experimental Linguistics. Linguist. Issues Lang. Technol., 2011, vol. 6, no. 0, pp. 1-27.

14. Zoltán S. and Jinil Y. Taxonomy, use cases, strengths and challenges of chat bots. Inf.Tarsad., Jul. 2018, vol. 18, no. 2, pp. 41-55. DOI: 10.22503/inftars.XVIII.2018.2.3.

15. Kiddon C., Zettlemoyer L., and Choi Y. Globally coherent text generation with neural checklist models. EMNLP 2016 - Conf. Empir. Methods Nat. Lang. Process. Proc., 2016, pp. 329-339. DOI: 10.18653/v1/d16-1032.

16. Writer B. Lithium-Ion Batteries. Cham: Springer International Publishing, 2019.

17. Pap G. and Szűcs G. Continuous Markov Chains; Kolmogorov Equations, in Stochastic Processes, 2013.

18. Ibidem.

19. Ogada K. and Mwangi W. N-gram Based Text Categorization Method for Improved Data Mining. J. Inf. Eng. Appl., 2015, vol. 5, no. 8, pp. 35-44.

20. Marco Romano, Luca Paolino, Genoveffa Tortora & Giuliana VitielloThe Tap and Slide Keyboard: A New Interaction Method for Mobile Device Text Entry, International Journal of Human – Computer Interaction, 2014, 30:12, 935-945, DOI: 10.1080/10447318.2014.924349.

21. Weir D., Pohl H., Rogers S., Vertanen K., and Kristensson P. O. Uncertain text entry on mobile devices. In Conference on Human Factors in Computing Systems - Proceedings, 2014, pp. 2307-2316. DOI: 10.1145/2556288.2557412.

22. Dey A. Machine Learning Algorithms: A Review. Int. J. Comput. Sci. Inf. Technol., 2016, vol. 7, no. 3, pp. 1174-1179.

23. Sebastiani F. Machine Learning in Automated Text Categorization. ACM Comput. Pressure.Mar. 2002, vol. 34, no. 1, pp. 1-47. DOI: 10.1145/505282.505283.

24. Yeo H. S., Phang X. S., Castellucci S. J., Kristensson P. O., and Quigley A. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. InConference on Human Factors in Computing Systems - Proceedings, May 2017, pp. 4194-4202. DOI: 10.1145/3025453.3025520.

25. Sotsenko A., Zbick J., Jansen M., and Milrad M. Flexible and contextualized cloud applications for mobile learning scenarios.InAdvances in Intelligent Systems and Computing, 2016, vol. 406, pp. 167-92.

26. Inan H. A. et al. Training Data Leakage Analysis in Language Models, 2021.

27. Askell A. et al. Better Language Models and Their Implications. OpenAI, 2019. Available at: https://openai.com/blog/better-language-models/ (accessed 12.11.2021).

28. Ibidem.

29. Wu X. and Lode M. [GPT-2] Language Models are Unsupervised Multitask Learners. Open AI Blog, May 2020, vol. 1, pp. 1-7.

30. Askell A. et al. Better Language Models and Their Implications. Open AI, 2019. Available at: https://openai.com/blog/better-language-models/ (accessed 2.11.2021).

31. Lai I. Conditional Text Generation by Fine Tuning GPT-2. Towards Data Science, 2021. Available at: https://towardsdatascience.com/conditional-text-generation-by-fine-tuning-gpt-2-11c1a9fc639d (accessed 19.11.2021).

32. Google-research/bert. Available at: https://github.com/google-research/bert (accessed 11.11.2021).

33. Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, Jingjing Liu. Distilling the Knowledge of BERT for Text Generation. ICLR 2020 Conference Withdrawn Submission, 2019.

34. Omeiza D., Adewole K. S., and Nkemelu D. EEG-based Communication with a Predictive Text Algorithm. Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2018). ArXiv, Nips, 2018.

35. America M. Talk to Transformer. The Routledge Handbook of Remix Studies and Digital Humanities, 2021. Availableat:https://app.inferkit.com/dem (accessed 08.11.2021).

Шандор Агоштон Пол, студент
Ижевский государственный технический университет имени М. Т. Калашникова, Ижевск, Россия
Факультет машиностроения, информатики и электротехники, Университет Святого Иштвана в Дьёре, Дьёр, Венгрия

«Невозможно отвергать, отрицать прогресс, этого никогда не было и никогда не будет, он необратим. Нужно научиться управлять искусственным интеллектом»
Владимир Владимирович Путин

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, СПОСОБНЫЕ ГЕНЕРИРОВАТЬ ДЛИННЫЕ ТЕКСТЫ С МИНИМАЛЬНЫМ НАЧАЛЬНЫМ ВВОДОМ

Статья посвящена описанию работы и исследованию генераторов текста для создания текстов большого объема с минимальным вводом первичной информации. Исследование носит междисциплинарный характер и выполнено на стыке обработки естественного языка и компьютерной лингвистики. Его актуальность связана с тем, что появление роста текстовой информации через интернет, значительно ускорившее развитие некоторых научных областей, существовавших на протяжении многих десятилетий, происходило не так быстро, как в последние десятилетия, в отсутствие большого доступа к входным данным. Во многих используемых системах, в том числе коммерческих, было предложено и реализовано несколько многообещающих идей и экспериментов в этой области для автоматической обработки текстов на естественном языке. В данном исследовании предложено новое понимание существующих приложений компьютерной лингвистики и их работы в каждой теме и их сравнение. Конкретные приложения относятся к одной теме с незначительными различиями в области их работы и применения. В исследовании были также описаны методы нейролингвистического программирования в области искусственного интеллекта, основанные на понимании, что письменность является самой важной и отслеживаемой формой коммуникации. Искусственный интеллект и автоматизация лежат в основе идущей четвертой промышленной революции, следовательно, среди прочего, следует понимать важность векторизации данных, а также то, что любой абстрактный процесс может быть хорошо смоделирован с помощью математических процессов. В заключение были сделаны выводы о том, что за последние 20 лет значительно развивались и дополнялись различные методики и методы, причем новейшие текстовые генераторы почти превзошли рамки и качество текста, написанного человеком, однако это не означает, что в будущем люди будут читать только машинный текст. Интуиция и творчество по-прежнему требуют присутствия человека по сей день, машина обрабатывает набор информации, которую мы вводим, и обрабатывает различные шаблоны, а затем генерирует тексты с их помощью.

Ключевые слова: нейролингвистическое программирование; искусственный интеллект; четвертая промышленная революция; генеративный предварительно обученный преобразователь 2; стохастические модели.

Получено: 24.11.2021

For Citation

Sándor Ágoston Pál [AI Methods that Can Generate Longer Texts with Minimal Initial Input]. *Social'no-jekonomicheskoe upravlenie: teorija i praktika*, 2021, vol. 17, no. 4, pp. 109-121 (in Russ.). DOI 10.22213/2618-9763-2021-4-109-121.