International Journal of Research in E-learning
Vol. 9 (2), 2023, pp. 1–22

DOI 10.31261/IJREL.2023.9.2.04

Tetiana A. Vakaliuk
Zhytomyr Polytechnic State University, Ukraine
https://orcid.org/0000-0001-6825-4697

Oleksii V. Chyzhmotria
Zhytomyr Polytechnic State University, Ukraine
https://orcid.org/0000-0002-5515-6550

Svitlana O. Didkivska
Krakow University of Economics, Poland
https://orcid.org/0000-0002-4004-6631

Illia Linevych
Zhytomyr Polytechnic State University, Ukraine
https://orcid.org/0000-0002-8189-4856

Development of a Web Service for Creating Tests Based on Text Analysis Using Natural Language Processing Technologies

Abstract

The purpose of the work is to analyze models, natural language processing methods, and select modern technologies for training these models, as well as to develop a web service for creating tests based on text analysis using natural language processing technologies. The study considers methods and algorithms for intelligent data analysis to generate questions and correct and incorrect answers from the text. The authors justify the choice of a neural network for generating tests based on English and Ukrainian text, and characterize data sources for training. The study also describes the activity of the proposed model, which will serve as a basis for creating a web service. After a detailed review of these datasets, the necessary data for the experiment were extracted and transformed into a convenient format for use. The training algorithm for 6 models was designed and implemented, and valuable metrics were obtained after their training. Additionally, a server-side and web interface were developed to interact with each other.

K e y w o r d s: text analysis, natural language, natural language processing technologies, NLP, model

Introduction

To assess students' knowledge and skills at the end of a semester or upon completion of any topic, instructors often conduct a summative assessment. While preparing for this process, the instructor needs to analyze the covered educational material and, based on it, creates various questions with corresponding answers. It is also essential not to forget about the subsequent test review, which involves calculating the results for each student. The entire process is very monotonous, consuming a lot of energy and time, especially if the instructor has a large number of students. A similar chain of events also occurs during the evaluation of a company's employee professionalism or, conversely, during their internship.

Therefore, the relevance of the topic is determined by the need to automate the process of creating tests from textual material using natural language processing technologies. Further research is required on the issue of properly training neural models for generating questions, correct and incorrect answers based on an English and Ukrainian text.

Tests are one of the primary forms of consolidating knowledge. The more accurately they are selected, the more effectively the learning process can be implemented. An automated system for creating questions could improve their quality and increase their number, thereby covering more details and topics for the learning program. With this system, instructors or employers could generate questions based on covered material to assess the knowledge and skills of their students and workers.

At present, there are numerous languages and nationalities, complicating this task. To efficiently generate tests from a text in various languages, it is necessary to consider the lexical, grammatical, and other features of each language. That is, a separate implementation plan must be developed for each language to achieve the best results. Therefore, to simplify the work, this project will focus on generating questions, correct and incorrect answer options from an English text, as it is one of the most widespread languages globally. Additionally, based on the conducted research, algorithms will be adapted for Ukrainian texts.

In general, all types of questions can be divided into two groups: objective and subjective. When answering objective questions, users are asked to choose the correct answer from the provided options. The most popular of this type are multiple-choice tasks, matching, true or false, filling in blanks in a sentence, and more. At the same time, when answering subjective questions, students need to write the answer independently. Tasks of this type may require a short response (from a few words to several sentences) or a long one (for example, an essay). Therefore, for a successful generation, one needs to have such a data pair: a text passage (minimum 2–3 sentences); a predetermined answer or a set of answers, based on which questions will be asked and incorrect options will be generated.

To avoid doing this manually, there are numerous methods for extracting so-called "keywords" from the text that can become potential answers, as well as algorithms that can generate false options based on these words. Using this, a comprehensive test can be created. One such interesting approach is Named Entity Recognition (NER). Named Entity Recognition is a subtask of information extraction that attempts to find and classify named entities in an unstructured text into predefined categories, such as names of people, organizations, places, time, quantities, monetary values, percentages, and more. That is, some named entities can be used as answers, and some of them, which match a category, can be used as incorrect options.

As for the generation of the questions themselves, this task is much more complex. Unfortunately, most static methods do not show very high efficiency and work well only with a few languages, such as English, while they cannot form a long, consistently correct chain of words for Ukrainian. Therefore, nowadays, artificial intelligence is increasingly replacing these algorithms, as this development allows for obtaining more accurate results, similar to the ones obtained by human. Considering this, further research was conducted in this area.

The **purpose** of the work is to analyze models, natural language processing methods, and select modern technologies for training these models, as well as to develop a web service for creating tests based on text analysis using natural language processing technologies.

Methodology of Research

To achieve this goal, the following main **tasks** need to be solved:

- define the basic requirements and desired end results;
- research existing neural models of natural language processing that are suitable for achieving the selected goals;
- select suitable models and methods of their training;
- make a reasonable choice of tools for model training;
- create an API to interact with models (server side);

- create a web interface for testing and general use of models (client side).

To solve these tasks, a number of methods were used: analysis, synthesis, and generalization to study neural models of natural language processing and to select tools for training models; and methods for training neural models using programming languages.

Theoretical Background

Data mining methods and algorithms for generating questions, correct and incorrect answers from the text

To implement the given task, natural language processing technologies can be used. Natural Language Processing (NLP) is the ability of a computer program to understand a human language. NLP is a branch of artificial intelligence that deals with the interaction between computers and humans using a natural language. The ultimate goal of NLP is to teach a neural network to read, decode, understand, and determine the meaning of human language in a valuable way. Most NLP techniques rely on various machine learning methods to derive meaning from a text (Education Ecosystem, 2022).

NLP is used in various development areas, namely: in language translation programs such as Google Translate; in text processors like Microsoft Word and Grammarly for checking the grammatical accuracy of texts; in interactive voice response programs; in personal assistants such as Google, Siri, Cortana, Alexa, and more.

There are two main stages of natural language processing: preprocessing of data and the development of the necessary algorithm to solve the given problem. Preprocessing of data includes preparing and “cleaning” text data for machines to be able to analyze it. Preprocessing presents the data in a workable form and highlights specific words in the text that the algorithm can work with. This stage may include the following processes (Lutkevich, 2022):

- tokenization – the text is broken down into smaller, separate units for processing;
- removal of stop words – common, frequently repeated words and parts of speech are removed from the text, leaving only unique ones that provide the most information about the text;
- lemmatization and stemming – words are reduced to their root forms for processing;
- part-of-speech tagging – words are marked with the part of speech they correspond to, such as nouns, verbs, and adjectives.

After preprocessing the data, the development of the text processing algorithm begins. There are many different natural language processing algorithms, but usually, two main types are used: rule-based systems – these systems use carefully pre-developed linguistic rules; machine learning-based systems – machine learning algorithms are used to train neural networks. They learn to perform tasks based on the training data provided to them and adjust their methods as they process more data (Lutkevich, 2022). The second approach will be used in this study.

Effective question generation is an open scientific problem that still lacks a universal, easy approach to its solution. The question generation itself consists of two fairly complex processes: analysis of the lexical and syntactic context of input sentences, and construction of new sentences based on the processed information, following all the rules of the current language. Therefore, after analyzing various sources, several works were found on the autogeneration of questions from a text in Turkish (Akyon et al., 2022), Arabic (Nagoudi et al., 2022), and English languages (Vachev et al., 2022).

In each of these works, special transformer models for natural language processing based on machine learning were used. A transformer (neural network) is a deep learning model. It uses an "attention" mechanism that takes into account the relationship between all the words in a sentence. The transformer creates differential coefficients that indicate the elements in the sentence that are most important for a more accurate interpretation of the meaning of problematic words. Thus, the computer can quickly and efficiently understand ambiguous phrases (Negri, 2022). Other scholars (Mellah et al., 2021; Affolter et al., 2019, Guo et al., 2019; Xavier et al., 2022) have also considered various aspects of the issue in their works.

Transformers can differ in the type of input and output data. For example, they can accept data in a text, visual, or audio format and return results of the same type. To solve the problem of generating tests, transformers that can work with text data and return a textual result will be considered. They are also called seq2seq models (sequence to sequence).

Seq2Seq models consist of two parts: an encoder and a decoder. The encoder and decoder can be thought of as translators who can speak only two languages. Each of them has its own native language; for example, one can say that the encoder is a native speaker of Chinese, and the decoder is a native speaker of English. Both have a second common language; let us say Japanese. To translate Chinese to English, the encoder converts the Chinese sentence into Japanese. This Japanese sentence is then passed as a context to the decoder. Since the decoder understands Japanese and can read in this language, it can now translate the given Japanese passage into English.

Another key component of the transformer architecture is a mechanism called "attention." This technique mimics cognitive attention. Cognitive attention reflects how our brain focuses on the meaningful parts of a sentence, helping us understand its overall meaning. For example, when you read this sentence, you are always focused on the word you are reading, but at the same time, your memory stores the most important keywords you have already read to ensure understanding of the context.

The attention mechanism examines the input text sequence in parts, and at each step, it decides which other parts of the sequence are important. This helps the transformer filter out noise and focus on what is relevant, connecting related words that, by themselves, do not contain any obvious markers pointing to one another.

Selecting a neural network for generating tests based on English and Ukrainian text

At present, there are many ready-made natural language processing models, each of which has been pre-configured on large datasets to be able to perform basic NLP tasks. During this initial training, transformers learn various language constructs and basic functions, which are sufficient to avoid having to train them from scratch. Thus, they begin to "understand" a human language. Datasets represent a large "repository" of records that are stored in a special format. By receiving these records as input, transformers learn to perform specific tasks. Pre-trained models are most often used in the implementation of NLP tasks because they are easier to adjust, have high accuracy, and require much less time and computational resources for additional training compared to custom transformers built from scratch (neural networks). Although most pre-trained transformers can perform a range of simple tasks, none of them can initially generate questions, correct or incorrect answers based on the analyzed text. Therefore, it is necessary to choose an NLP model for each task and train them separately. There are countless neural networks that can work with texts, but the choice of the best is limited by a number of factors, namely:

1. The ability of the neural network to work with texts in different languages.
2. The amount of computational resources needed for training and operation of the neural network.
3. The availability of the neural network on the Internet.

After analyzing several natural language processing models, the most optimal option was found to be the Text-To-Text Transfer Transformer (T5) developed by Google (Roberts, 2022). In T5, all NLP tasks are transformed into a unified text format in which the input and output data are always text strings, unlike models like BERT (Devlin, 2019), which accept a text as an input but return results as quantitative estimates.

The T5 transformer was pre-trained on a large dataset, the Colossal Clean Crawled Corpus (C4) (Dodge et al., 2021), so that it understands a human language and can perform a range of tasks, such as data classification or text translation. An important component for further training of neural networks for more specific tasks is the unlabeled dataset used for pre-training. To enhance the pre-training effect, a dataset is needed that will not only be high-quality and diverse, but also massive. Existing datasets for pre-training do not meet all three of these criteria – for example, a collection of articles from Wikipedia is of high quality but is unified in style and relatively small in size, while the Common Crawl dataset (Common crawl) is huge and very diverse but has a relatively low quality.

So, to satisfy all requirements, Google developed the Colossal Clean Crawled Corpus (C4), a cleaned version of Common Crawl, which is two orders of magnitude larger than the collection of Wikipedia articles. The cleaning process involved deduplication, discarding incomplete sentences, and removing offensive or noisy content. This filtering led to better initial training results for the T5 model. As for the language, the T5 model works only with English, but there is its multilingual version (mT5), which can be used for Ukrainian. This model also has different versions (dimensions), which allows it to be used even on ordinary machines with a single graphics processor. It is worth noting that the smaller the size of the neural network, the less accurate the results it produces, but their quality remains satisfactory. Such models work much faster and are more accessible to ordinary users.

To train the T5 model for generating questions, correct and incorrect answers, special datasets need to be used. Also, in training the transformer, the optimization function plays an important role, helping to reduce the error of future calculations. Mathematical optimization (sometimes referred to as optimization) or mathematical programming in mathematics, computer science, and operations research refers to the selection of the best element (according to a certain criterion) from a set of available alternatives. In the simplest case, the optimization task consists of finding the extremum (minimum or maximum) of a real function by systematically selecting input values from the allowed set and calculating the function's value. The optimization function to be used is the AdamW function (Graetz, 2022), which is an improved version of the Adam method.

Comparison of available software products

The use of artificial intelligence is achieving great success in the field of education. A striking example of such development is the increase in the number of online platforms and mobile applications for teaching. Machine learning, deep learning, and natural language processing are used to assist teachers in their work and help students increase their productivity.

Let us consider software products with the ability to auto-generate questions and determine the features of each of them: PrepAI, Quillionz, and Questgen. For better perception, we will divide the analysis into certain evaluation criteria:

1. Data submission methods, input formats, and sources

The first factor is the data input method. PrepAI allows multiple input formats. It can read data from plain text, files (.pdf, .docx), video files or YouTube videos, as well as Wikipedia articles. Quillionz also supports multiple input data formats: plain text, ".pdf" files, and YouTube videos. In addition, Quillionz allows you to choose a topic before generating questions, which helps the algorithm perform better. Questgen, on the other hand, can only work with a text. Unfortunately, each of the services supports only the English language, which is a significant drawback and emphasizes the relevance of conducting research on test generation for the Ukrainian language as well.

2. Types of questions and their generation

Generation in all applications is relatively fast. The process mainly takes from a few seconds to a minute, but this number depends on the amount of input data.

3. Output data and storage

After the questions have been generated, they can be edited and saved. PrepAI and Quillionz allow for editing and deleting questions with answers, while Questgen, in addition, provides the ability to change their order. Unfortunately, none of the services allows adding your own questions to the final result. In PrepAI, you can export the created test in ".pdf", ".docx", or ".xlsx" format. Quillionz allows for exporting data in ".txt", ".pdf", and ".docx" format. As for Questgen, it does not allow for exporting the resulting test, but it can be copied as a text, saved to a personal account, and printed in the prepared format. From the analysis, it can be said that PrepAI mainly wins in terms of functionality, but Questgen showed the best results in terms of the quality of generated questions.

Results

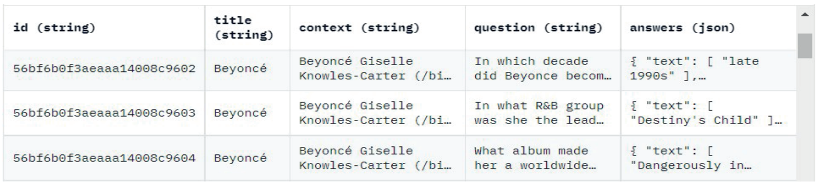
The server-side implementation, data processing, and training will be conducted using the Python programming language, as it contains numerous built-in methods and libraries that greatly simplify the development process. The model training will take place in the Google Colaboratory cloud development environment (Google Colaboratory, 2022). A single-page application (SPA) type of website is chosen for a web interface development. The web application developed following this scheme uses a single page and allows for dynamically changing content without reloading, providing the user with an experience similar to using a native application. The web interface will be written using the TypeScript programming language and the React library.

Characteristics of data sources for training

For the task of generating questions, the Stanford Question Answering Dataset (SQuAD) can be utilized. SquAD is a reading comprehension dataset consisting of questions posed on English Wikipedia articles. The answer to each question is a word or text snippet from the corresponding passage or the question can be unanswerable. Generally, SquAD was created for the opposite task – generating answers to questions, but it is well-suited for the current task.

There are two versions of this dataset: SquAD 1.1 – the previous version of the SquAD dataset, containing approximately 100,000 question-answer pairs based on 500+ articles; SquAD 2.0 – combines questions from SquAD 1.1 with unanswerable questions written by crowdworkers to resemble answerable ones. SquAD 2.0 will be used for training since this dataset contains two types of questions, allowing the model to learn to generate questions based on both predefined answers and without them.

Now let us take a closer look at the structure of the selected dataset. Figure 1 shows several entries.

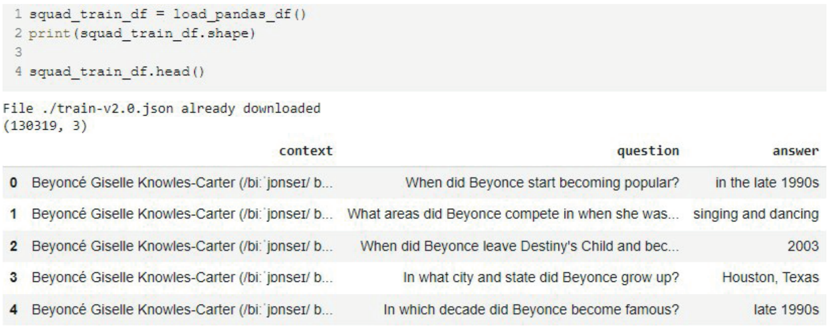


id (string)	title (string)	context (string)	question (string)	answers (json)
56bf6b0f3aeaaa14008c9602	Beyoncé	Beyoncé Giselle Knowles-Carter (/bi…	In which decade did Beyonce becom…	{ "text": ["late 1990s"],…
56bf6b0f3aeaaa14008c9603	Beyoncé	Beyoncé Giselle Knowles-Carter (/bi…	In what R&B group was she the lead…	{ "text": ["Destiny's Child"]…
56bf6b0f3aeaaa14008c9604	Beyoncé	Beyoncé Giselle Knowles-Carter (/bi…	What album made her a worldwide…	{ "text": ["Dangerously in…

Figure 1. SquAD 2.0 structure

S o u r c e: Own work.

In fact, the dataset is split into two files: train and dev. This is done for convenience. The first file is used for model training, while the second is used for validation after each training epoch. The SquAD 2.0 translation into Ukrainian has already been carried out in a fellow researcher's work (Huggingface.co, 2022), where the dataset was used for generating answers to questions. For the task at hand, it is sufficient to have only a context, question, and answer. Moreover, the data is stored in the less convenient JSON format rather than CSV. So let us modify the dataset to meet our needs. For entries without an answer, the answer column will contain a special token "[MASK]" to train the model to generate questions based only on the context. Figures 2 and 3 show the result of the data transformation.



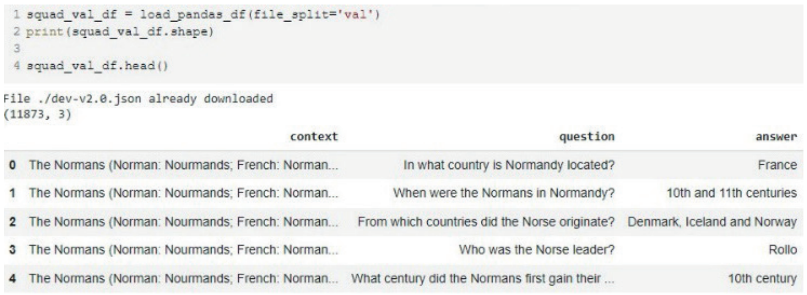
```python
squad_train_df = load_pandas_df()
print(squad_train_df.shape)

squad_train_df.head()
```

```
File ./train-v2.0.json already downloaded
(130319, 3)
```

	context	question	answer
0	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	When did Beyonce start becoming popular?	in the late 1990s
1	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	What areas did Beyonce compete in when she was...	singing and dancing
2	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	When did Beyonce leave Destiny's Child and bec...	2003
3	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	In what city and state did Beyonce grow up?	Houston, Texas
4	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	In which decade did Beyonce become famous?	late 1990s

Figure 2. Modified training set for generating questions

S o u r c e: Own work.



```python
squad_val_df = load_pandas_df(file_split='val')
print(squad_val_df.shape)

squad_val_df.head()
```

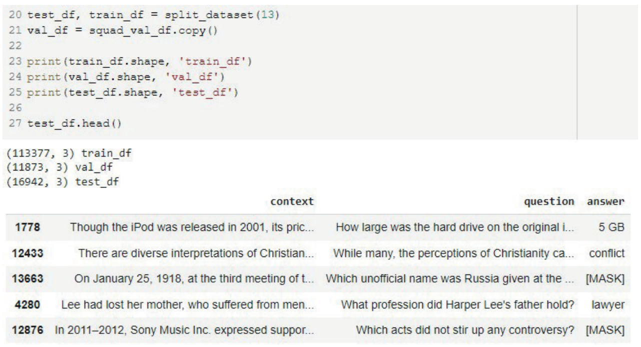
```
File ./dev-v2.0.json already downloaded
(11873, 3)
```

	context	question	answer
0	The Normans (Norman: Nourmands; French: Norman...	In what country is Normandy located?	France
1	The Normans (Norman: Nourmands; French: Norman...	When were the Normans in Normandy?	10th and 11th centuries
2	The Normans (Norman: Nourmands; French: Norman...	From which countries did the Norse originate?	Denmark, Iceland and Norway
3	The Normans (Norman: Nourmands; French: Norman...	Who was the Norse leader?	Rollo
4	The Normans (Norman: Nourmands; French: Norman...	What century did the Normans first gain their ...	10th century

Figure 3. Modified validation set for generating questions

S o u r c e: Own work.

Now, let us set aside a few percent of the training dataset for the test set, which will be used for checking the results after training. Figure 4 shows the result.



```python
test_df, train_df = split_dataset(13)
val_df = squad_val_df.copy()

print(train_df.shape, 'train_df')
print(val_df.shape, 'val_df')
print(test_df.shape, 'test_df')

test_df.head()
```

```
(113377, 3) train_df
(11873, 3) val_df
(16942, 3) test_df
```

	context	question	answer
1778	Though the iPod was released in 2001, its pric...	How large was the hard drive on the original i...	5 GB
12433	There are diverse interpretations of Christian...	While many, the perceptions of Christianity ca...	conflict
13663	On January 25, 1918, at the third meeting of t...	Which unofficial name was Russia given at the ...	[MASK]
4280	Lee had lost her mother, who suffered from men...	What profession did Harper Lee's father hold?	lawyer
12876	In 2011–2012, Sony Music Inc. expressed suppor...	Which acts did not stir up any controversy?	[MASK]

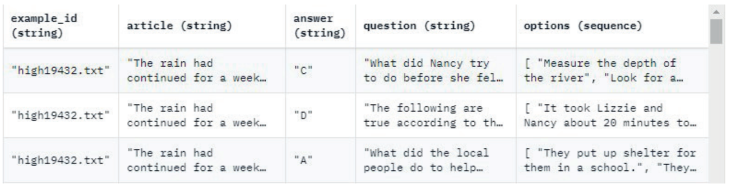
Figure 4. Creating a test data set for question generation

S o u r c e: Own work.

For the task of extracting answers from the text, the SQuAD dataset will also be used, but we will transform the data so that each entry contains the context and a list of potential answers found within it. Splitting into three separate datasets followed a similar principle as in the previous example.

To perform the task of generating incorrect answers, the ReAding Comprehension dataset from Examinations (RACE) can be used. RACE is a dataset containing more than 28,000 paragraphs and approximately 100,000 questions. This dataset is based on English language exams designed for middle and high school students in China. Unlike the SQuAD dataset, which contains only questions and answers, RACE also includes a list of incorrect answer options for each question-answer pair. Now, let us take a closer look at the structure of the chosen dataset. Figure 5 shows several entries.

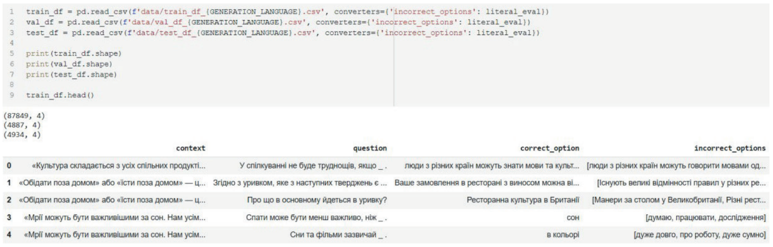
The dataset is already divided into three files: train, dev, and test. The translation of RACE into Ukrainian was carried out independently. Figure 6 shows the transformed datasets in a convenient format for training.



example_id (string)	article (string)	answer (string)	question (string)	options (sequence)
"high19432.txt"	"The rain had continued for a week…	"C"	"What did Nancy try to do before she fel…	["Measure the depth of the river", "Look for a…
"high19432.txt"	"The rain had continued for a week…	"D"	"The following are true according to th…	["It took Lizzie and Nancy about 20 minutes to…
"high19432.txt"	"The rain had continued for a week…	"A"	"What did the local people do to help…	["They put up shelter for them in a school.", "They…

Figure 5. RACE structure

S o u r c e: Own work.



```
train_df = pd.read_csv(f'data/train_df_{GENERATION_LANGUAGE}.csv', converters={'incorrect_options': literal_eval})
val_df = pd.read_csv(f'data/val_df_{GENERATION_LANGUAGE}.csv', converters={'incorrect_options': literal_eval})
test_df = pd.read_csv(f'data/test_df_{GENERATION_LANGUAGE}.csv', converters={'incorrect_options': literal_eval})

print(train_df.shape)
print(val_df.shape)
print(test_df.shape)

train_df.head()
```

```
(87849, 4)
(4887, 4)
(4934, 4)
                                       context                                     question                               correct_option                               incorrect_options
0   «Культура складається з усіх спільних продукті...        У спілкуванні не буде труднощів, якщо _ .   люди з різних країн можуть знати мови та культ...   [люди з різних країн можуть говорити мовами од...
1   «Обідати поза домом» або «їсти поза домом» — ц...   Згідно з уривком, яке з наступних тверджень є ...   Ваше замовлення в ресторані з виносом можна ві...   [Існують великі відмінності правил у різних ре...
2   «Обідати поза домом» або «їсти поза домом» — ц...          Про що в основному йдеться в уривку?                    Ресторанна культура в Британії   [Манери за столом у Великобританії, Різні рест...
3   «Мрії можуть бути важливішими за сон. Нам усім...          Спати може бути менш важливо, ніж _ .                                          сон           [думаю, працювати, дослідження]
4   «Мрії можуть бути важливішими за сон. Нам усім...                 Сни та фільми зазвичай _ .                                     в кольорі        [дуже довго, про роботу, дуже сумно]
```

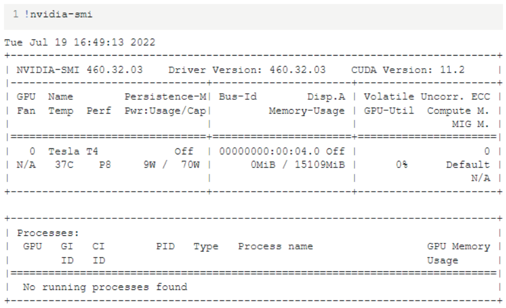
Figure 6. Modified datasets for generation of incorrect answer choices

S o u r c e: Own work.

Model training and practical implementation

For each of the three identified tasks, a separate model for both English and Ukrainian languages needs to be trained. Therefore, a total of 6 transformers will be trained. In each of the three cases, the overall training process is similar and differs in terms of some details. So, let us consider only the common methods.

We will start training the T5 model. It is best to conduct training on a graphics processor because it is a complex computational process that can take a lot of time, especially if using a regular CPU, which has far fewer cores. Google Colaboratory offers the opportunity to use powerful graphics processors for free but limits their continuous usage to 6 hours, which is also important to consider since sometimes training models can take much longer, so it has to be done in parts. To ensure that the training is performed on a GPU, go to Edit -> Notebook settings -> Hardware accelerator and select GPU. Check the information about the GPU chip to which the current notebook has an access (Figure 7).



```
!nvidia-smi

Tue Jul 19 16:49:13 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   37C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

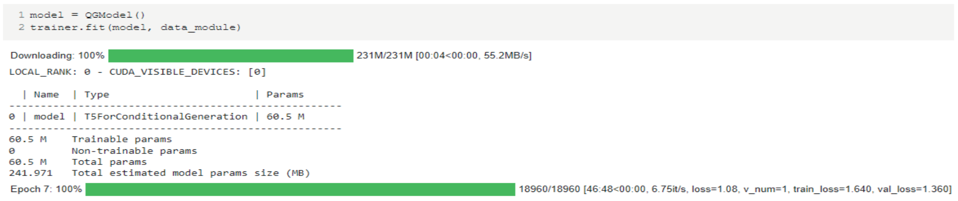
Figure 7. GPU information

Source: Own work.

Next, we choose the language for which we want to perform training and connect Google Drive to the runtime environment so that we can write and read files from it. Now, we set up Google Drive as the working directory for storing all checkpoints and logs.

The “transformers” library contains various NLP models for solving a wide range of tasks. In this work, we export the T5 model from it. The “pytorch-lightning” library contains many useful classes that simplify the training process and reduce the amount of a written code. The “tokenizers” library contains special tokenizers that prepare the text input for the model. We import the required methods and classes.

Next, we need to prepare the dataset for use. After that, appropriate classes with methods for training are created. For each task, the classes were developed separately, but their operating principle is quite similar. The following is an example of classes for generating questions. We inherit a special Dataset class, which manipulates the dataset. We also add another special token “SEP” to separate parts of input and output sequences. Now we create a DataModule class. DataModule is responsible for creating Datasets (for training, validation, and testing) and for creating appropriate DataLoaders for iterating over them. In the end, we create a Trainer, which will carry out the training, and a ModelCheckpoint that will save the model after each epoch. This way, the best model can be used later on. Figure 8 shows the training process.



```
model = QGModel()
trainer.fit(model, data_module)
```

```
Downloading: 100% 231M/231M [00:04<00:00, 55.2MB/s]
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name  | Type                       | Params
------------------------------------------------------
0 | model | T5ForConditionalGeneration | 60.5 M
------------------------------------------------------
60.5 M    Trainable params
0         Non-trainable params
60.5 M    Total params
241.971   Total estimated model params size (MB)
Epoch 7: 100% 18960/18960 [46:48<00:00, 6.75it/s, loss=1.08, v_num=1, train_loss=1.640, val_loss=1.360]
```

Figure 8. Training model

S o u r c e: Own work.

Now we can move on to the time metrics. Due to Google Colaboratory limitations, the training of some models took place in two or even three stages. However, the pytorch-lightning library can save logs and the state of the model being trained at the time of urgent and emergency disconnections from the runtime environment. For the question generation task, the process of training the English model took approximately 6 hours (8 epochs). Training the Ukrainian model took approximately 9 hours (12 epochs). For the potential answer extraction task, the process of training the English model took approximately 2 hours (6 epochs). Training the Ukrainian model took slightly over 2 hours (8 epochs). For the incorrect answer generation task, the process of training the English model took approximately 6 hours (10 epochs). Training the Ukrainian model took approximately 8 hours (8 epochs).

Results of the training

We load the trained models and test them on the test dataset. Now we install the “tensorboard” extension to display graphs with metrics and specify the path to the directory where logs were stored.

Below, in Figures 9–11, the main loss minimization graphs during training and validation for the English models are presented.

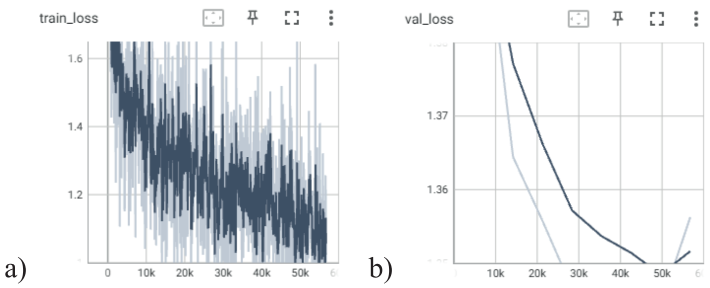


Figure 9. Graph of loss changes during a) training / b) validation for the task of generating questions

S o u r c e: Own work.

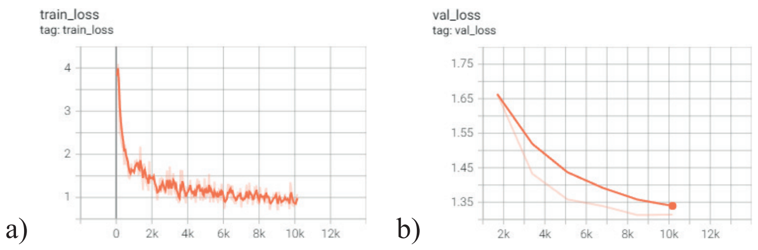


Figure 10. a) Graph of loss changes during training for the task of extracting potential responses. b) Graph of loss changes during validation for the task of extracting potential answers

S o u r c e: Own work.

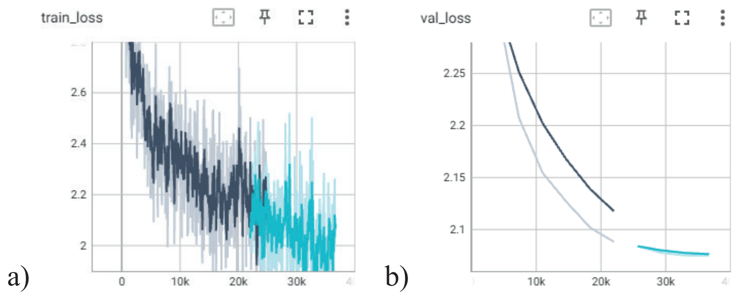


Figure 11. a) Graph of loss changes during training for the task of generating incorrect answers. b) Graph of loss changes during validation for generating incorrect answers

S o u r c e: Own work.

Now, in Figures 12–14, the main loss minimization graphs during training and validation for the Ukrainian models are presented.

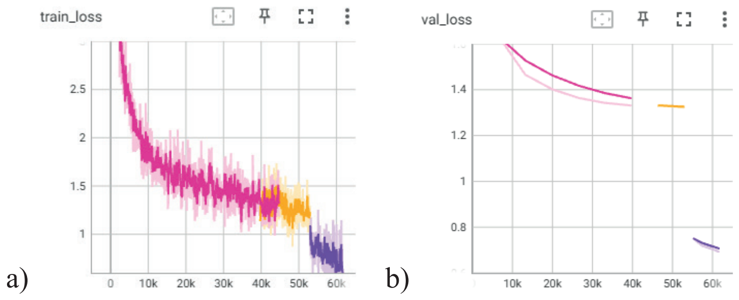


Figure 12. Ukrainian models graph of loss changes during a) training / b) validation for the task of generating questions

S o u r c e: Own work.

As can be seen from the figures, with each training epoch, the neural models provided more and more accurate results. Now we can move on to the demonstration of their use.

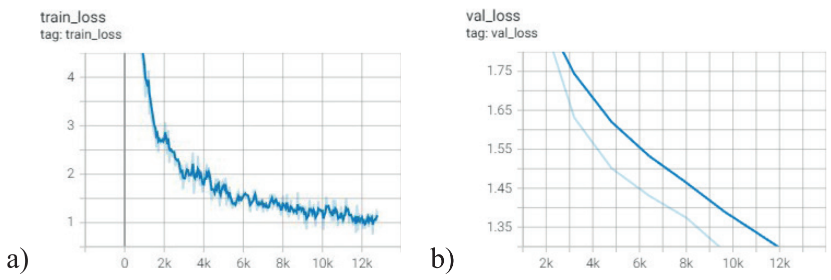


Figure 13. Ukrainian models a) Graph of loss changes during training for the task of extracting potential responses. b) Graph of loss changes during validation for the task of extracting potential answers

S o u r c e: Own work.

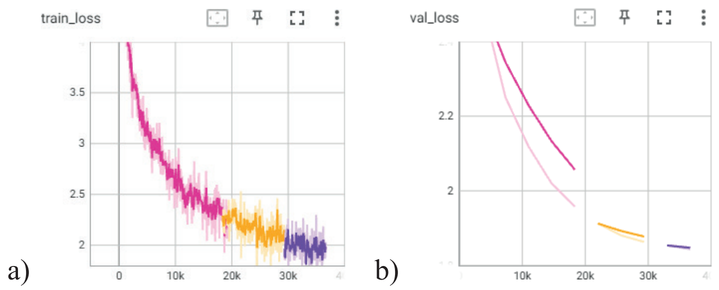
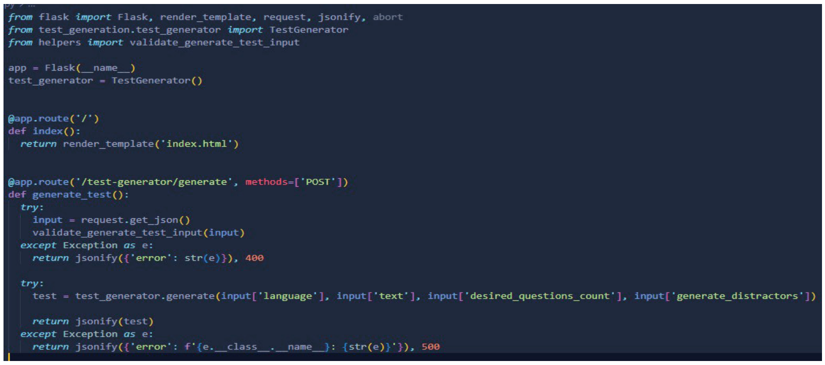


Figure 14. Ukrainian models a) Graph of loss changes during training for the task of extracting potential responses. b) Graph of loss changes during validation for the task of extracting potential answers

S o u r c e: Own work.

Development of the server API

After training, the models were saved to separate files, and a special API for their use was written using the Python programming language and the Flask library. Figure 16 shows the main endpoints.



```python
from flask import Flask, render_template, request, jsonify, abort
from test_generation.test_generator import TestGenerator
from helpers import validate_generate_test_input

app = Flask(__name__)
test_generator = TestGenerator()


@app.route('/')
def index():
  return render_template('index.html')


@app.route('/test-generator/generate', methods=['POST'])
def generate_test():
  try:
    input = request.get_json()
    validate_generate_test_input(input)
  except Exception as e:
    return jsonify({'error': str(e)}), 400

  try:
    test = test_generator.generate(input['language'], input['text'], input['desired_questions_count'], input['generate_distractors'])

    return jsonify(test)
  except Exception as e:
    return jsonify({'error': f'{e.__class__.__name__}: {str(e)}'}), 500
```

Figure 15. API endpoints

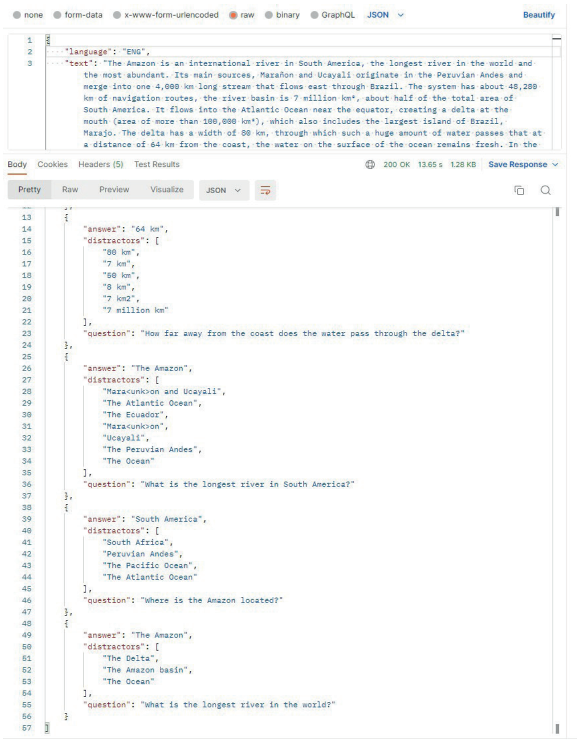
S o u r c e: Own work.

The first endpoint contains a description of the usage, and the second provides the access to the trained models, with which a complete test generation can be performed.

Let us look at the parameters accepted by the second server API endpoint in more detail:

- language – the input text language. Possible values: “ENG”, “UKR”;
- text – a paragraph of the text from which tests will be generated. A minimum paragraph character counts: 50;
- desired_questions_count – the desired number of questions to be generated. Possible values: from 1 to 20. Can be “null” if the “predefined_answers” parameter is specified;
- predefined_answers – answers chosen by the user. Possible values: str[], 1 to 20. Can be “null” if the “desired_questions_count” parameter is specified;
- generate_distractors – determines whether to generate incorrect answers. Possible values: “True”, “False”.

After sending a request for generation, the parameters undergo validation. If the validation is successful, the data enters the main class responsible for test generation and connects the previously trained models. First, the text is cleared of noise. Next, the filtered text is broken down into smaller parts. The spaCy library is responsible for this. It uses numerous natural language processing methods and is able to work with many languages. After this, one of the neural networks must find potential answers, the number of which should correspond to the desired number of generated questions. If the user enters potential answers themselves, this step is ignored. The next step is creating questions. Based on the found answers and the text snippets they relate to, another neural network generates questions. Finally, if the “generate_distractors” parameter contains the “True” value, the third neural model will begin generating incorrect answer options for each question.



```
{
    "language": "ENG",
    "text": "The Amazon is an international river in South America, the longest river in the world and
        the most abundant. Its main sources, Marañon and Ucayali originate in the Peruvian Andes and
        merge into one 4,000 km long stream that flows east through Brazil. The system has about 48,280
        km of navigation routes, the river basin is 7 million km², about half of the total area of
        South America. It flows into the Atlantic Ocean near the equator, creating a delta at the
        mouth (area of more than 100,000 km²), which also includes the largest island of Brazil,
        Marajo. The delta has a width of 80 km, through which such a huge amount of water passes that at
        a distance of 64 km from the coast, the water on the surface of the ocean remains fresh. In the
```

```
    {
        "answer": "64 km",
        "distractors": [
            "80 km",
            "7 km",
            "50 km",
            "8 km",
            "7 km2",
            "7 million km"
        ],
        "question": "How far away from the coast does the water pass through the delta?"
    },
    {
        "answer": "The Amazon",
        "distractors": [
            "Mara<unk>on and Ucayali",
            "The Atlantic Ocean",
            "The Ecuador",
            "Mara<unk>on",
            "Ucayali",
            "The Peruvian Andes",
            "The Ocean"
        ],
        "question": "What is the longest river in South America?"
    },
    {
        "answer": "South America",
        "distractors": [
            "South Africa",
            "Peruvian Andes",
            "The Pacific Ocean",
            "The Atlantic Ocean"
        ],
        "question": "Where is the Amazon located?"
    },
    {
        "answer": "The Amazon",
        "distractors": [
            "The Delta",
            "The Amazon basin",
            "The Ocean"
        ],
        "question": "What is the longest river in the world?"
    }
]
```

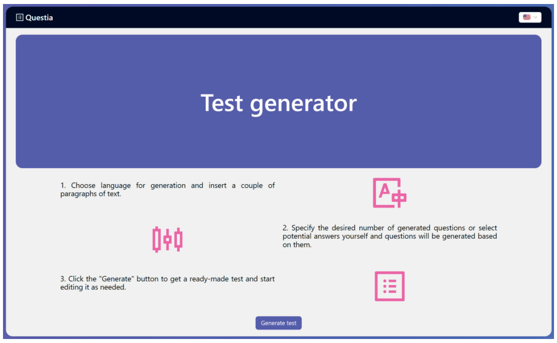
Figure 16. Generating questions based on the text in English

S o u r c e: Own work.

This model takes into account the text snippet on which the question was based, and adjacent ones to increase result diversity. In the final stage, the generated questions, correct, and incorrect answer options are formatted and combined into a single JSON object that is returned to the client. At this point, test generation can be considered complete. The test creation time varies greatly and depends on the desired number of questions and the need to generate correct (the user can specify them independently) or incorrect answers. On average, generating a test with 15–20 questions takes about two minutes. Figure 16 shows the testing of the server API and the generation of five questions from a paragraph of a text. All requests were executed using the Postman program.

Web interface

As mentioned earlier, a web application was developed for using the server API using the TypeScript programming language and the React library. Let us consider its structure and capabilities. Figure 17 shows the main page with instructions for use. The application supports two interface languages: English and Ukrainian. To change it, click on the selector in the upper right corner and choose the desired option.



1. Choose language for generation and insert a couple of paragraphs of text.

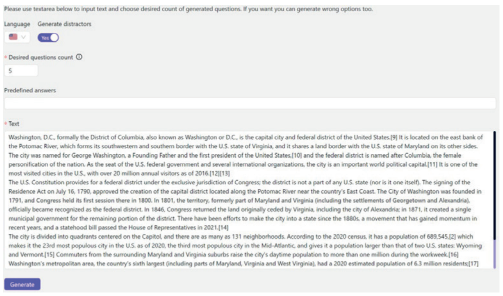
2. Specify the desired number of generated questions or select potential answers yourself and questions will be generated based on them.

3. Click the "Generate" button to get a ready-made test and start editing it as needed.

Figure 17. Web application home page

S o u r c e: Own work.

All further instructions will be executed for the English interface. To start creating a test, click the "Generate test" button. After that, you will be redirected to the generation form page. This form contains fields that correspond to the server API parameters: Language – the language of the text; Generate distractors – a checkbox indicating whether to generate incorrect answer options; Desired questions count – the desired number of generated questions. This is a required field, but if potential answers (Predefined answers) are specified manually, this field is ignored; Predefined answers – desired answers specified manually, based on which questions will be generated. Answers must be separated by a semicolon ";"; Text – the text from which the test will be generated. Fill in the form and try to generate a test with 5 questions (Figure 18, a). Click the "Generate" button to get the result. Now you can start editing the test in a special text editor (Figure 18, b). The answers are highlighted in a bold format.

a) 

Please use textarea below to input text and choose desired count of generated questions. If you want you can generate wrong options too.

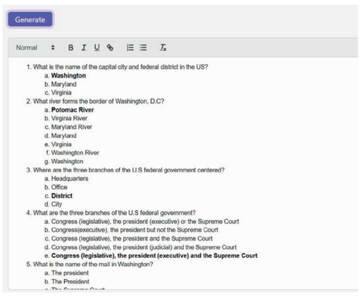
Language: 🇺🇸
Generate distractors: Yes

* Desired questions count: 5

Predefined answers:

* Text:

Washington, D.C., formally the District of Columbia, also known as Washington or D.C., is the capital city and federal district of the United States.[9] It is located on the east bank of the Potomac River, which forms its southwestern and southern border with the U.S. state of Virginia, and it shares a land border with the U.S. state of Maryland on its other sides. The city was named for George Washington, a Founding Father and the first president of the United States,[10] and the federal district is named after Columbia, the female personification of the nation. As the seat of the U.S. federal government and several international organizations, the city is an important world political capital.[11] It is one of the most visited cities in the U.S., with over 20 million annual visitors as of 2016.[12][13]
The U.S. Constitution provides for a federal district under the exclusive jurisdiction of Congress; the district is not a part of any U.S. state (nor is it one itself). The signing of the Residence Act on July 16, 1790, approved the creation of the capital district located along the Potomac River near the country's East Coast. The City of Washington was founded in 1791, and Congress held its first session there in 1800. In 1801, the territory, formerly part of Maryland and Virginia (including the settlements of Georgetown and Alexandria), officially became recognized as the federal district. In 1846, Congress returned the land originally ceded by Virginia, including the city of Alexandria; in 1871, it created a single municipal government for the remaining portion of the district. There have been efforts to make the city into a state since the 1880s, a movement that has gained momentum in recent years, and a statehood bill passed the House of Representatives in 2021.[14]
The city is divided into quadrants centered on the Capitol, and there are as many as 131 neighborhoods. According to the 2020 census, it has a population of 689,545,[2] which makes it the 23rd most populous city in the U.S. as of 2020, the third most populous city in the Mid-Atlantic, and gives it a population larger than that of two U.S. states: Wyoming and Vermont.[15] Commuters from the surrounding Maryland and Virginia suburbs raise the city's daytime population to more than one million during the workweek.[16] Washington's metropolitan area, the country's sixth largest (including parts of Maryland, Virginia and West Virginia), had a 2020 estimated population of 6.3 million residents;[17]

b) 

1. What is the name of the capital city and federal district in the US?
   a. **Washington**
   b. Maryland
   c. Virginia
2. What river forms the border of Washington, D.C?
   a. **Potomac River**
   b. Virginia River
   c. Maryland River
   d. Maryland
   e. Virginia
   f. Washington River
   g. Washington
3. Where are the three branches of the U.S federal government centered?
   a. Headquarters
   b. Office
   c. **District**
   d. City
4. What are the three branches of the U.S federal government?
   a. Congress (legislative), the president (executive) or the Supreme Court
   b. Congress(executive), the president but not the Supreme Court
   c. Congress (legislative), the president and the Supreme Court
   d. Congress (legislative), the president (judicial) and the Supreme Court
   e. **Congress (legislative), the president (executive) and the Supreme Court**
5. What is the name of the mall in Washington?
   a. The president
   b. The President

Figure 18. a) Generation of a five-question test; b) Generated test from English text

S o u r c e: Own work.

Discussion

The problem of auto-generating questions and correct and incorrect answers based on a particular text is relevant and widespread for the research. This will make it easier for teachers to create questions and answer options in the future. The proposed approach can be easily adapted to other languages, not just English.

In the future, teachers should be encouraged to try using such an application for educational purposes and analyze its advantages and disadvantages according to various criteria and indicators. Developing such criteria and indicators involves using the expert research method, which will be the next stage of the study.

Conclusions

In this study, the problem of autogeneration of questions, correct and incorrect answers based on the English text was identified. Based on this, several scientific works on this topic were researched and analyzed. As a result, it was decided to create a proprietary algorithm that would perform this task, i.e., generate questions, correct and incorrect answers from an English text. Special SQuAD and RACE datasets were chosen for training. After a detailed review of these datasets, the necessary data for the experiment were extracted and transformed into a convenient format for use. The training algorithm for 6 models was designed and implemented, and useful metrics were obtained after their training. Additionally, a server-side and web interface were developed for interaction with them.

In conclusion, the models were configured correctly, and now they fully perform the assigned task. Overall, the conducted research and work can be considered successful, as the web application requires a minimal amount of user actions.

The prospects for further research include studying the possibility and feasibility of using this software tool for educational purposes.

References

Affolter, K., Stockinger, K. & Bernstein, A. (2019). A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5), 793-819. https://doi.org/10.1007/s00778-019-00567-8.

Akyon, F., Cavusoglu, D., Cengiz, C., Altinuc, S. & Temizel, A. (2022) Automated question generation and question answering from Turkish texts using text-to-text transformers. *Turkish Journal of Electrical Engineering and Computer Sciences* (30:5), article 17. https://doi.org/10.55730/1300-0632.3914.

Colab.research.google.com. (2022) Google Colaboratory. Retrieved: https://colab.research.google.com/notebooks/welcome.ipynb?hl=ua.

Common crawl. https://commoncrawl.org/.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran & T. Solorio (Eds.). Proceedings of NAACL-HLT 2019, (pp. 4171–4186). https://aclanthology.org/N19-1423.pdf.

Dodge, J. et al. (2021) Documenting large Webtext corpora: A case study on the Colossal Clean Crawled Corpus, Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, (pp. 1286–1305). https://aclanthology.org/2021.emnlp-main.98.pdf.

Education Ecosystem (LEDU). (2022) A Simple Introduction to Natural Language Processing, Becoming Human: Artificial Intelligence Magazine. https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32.

Graetz, F. M. (2022). Why AdamW matters, Towards Data Science. https://towardsdatascience.com/why-adamw-matters-736223f31b5d.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J. G. & Liu, T. (2019). Towards complex text-to-SQL in cross-domain database with intermediate representation. In 57th Annual Meeting of the Association for Computational Linguistics, (pp. 4524–4535), Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1444.

Huggingface.co. (2022) squad_v2 · Datasets at Hugging Face. https://huggingface.co/datasets/squad_v2.

Lutkevich, B. (2022) What is Natural Language Processing? An Introduction to NLP. https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP.

Mellah, Y., Rhouati, A., Ettifouri, E. H., Bouchentouf, T. & Belkasmi, M. G. (2021). SQL Generation from Natural Language: A Sequence-to-Sequence Model Powered by the Transformers Architecture and Association Rules. *Journal of Computer Science*, 17(5), 480–489. https://doi.org/10.3844/jcssp.2021.480.489.

Nagoudi, E., Elmadany, A. & Abdul-Mageed, M. (2022) AraT5: Text-to-Text Transformers for Arabic Language Understanding and Generation. In S. Muresan, P. Nakov, & A. Villavicencio (Eds.). Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), (pp. 628–647). http://dx.doi.org/10.18653/v1/2022.acl-long.47.

Negri, D. (2022). Transformer NLP explained & natural language processing examples, Eidosmedia. https://www.eidosmedia.com/blog/technology/machine-learning-size-isn-t-everything.

Roberts, A. (2022). Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer. Google Research. https://ai.googleblog.com/2020/02/exploring-transfer-learn ing-with-t5.html.

Vachev, K., Hardalov, M., Karadzhov, G., Georgiev, G., Koychev, I. & Nakov, P. (2022). Leaf: Multiple-Choice Question Generation. Advances in Information Retrieval. ECIR 2022. Lecture *Notes in Computer Science* (13186). Springer, Cham, (pp. 321–328). https://doi.org/10.1007/978-3-030-99739-7_41.

Xavier, B. A. & Chen, P. H. (2022) Natural Language Processing for Imaging Protocol Assignment: Machine Learning for Multiclass Classification of Abdominal CT Protocols Using Indication Text Data. *Journal of Digital Imaging* 35, 1120–1130. https://doi.org/10.1007/s10278-022-00633-8.

Tetiana A. Vakaliuk, Oleksii V. Chyzhmotria, Svitlana O. Didkivska, Illia Linevych

Opracowanie usługi internetowej do tworzenia testów opartych na analizie tekstu z wykorzystaniem technologii przetwarzania języka naturalnego

Streszczenie

W artykule przeanalizowano modele i metody przetwarzania języka naturalnego oraz wybrano nowoczesne technologie szkolenia modeli w celu opracowania usługi internetowej do tworzenia testów opartych na analizie tekstu z wykorzystaniem technologii przetwarzania języka naturalnego. Badanie uwzględnia metody i algorytmy inteligentnej analizy danych w celu generowania pytań oraz poprawnych i niepoprawnych odpowiedzi z tekstu. Autorzy uzasadniają wybór sieci neuronowej do generowania testów na podstawie tekstu w języku angielskim i ukraińskim oraz charakteryzują źródła danych do szkolenia. Badanie opisuje również działanie proponowanego modelu, który posłuży jako podstawa do stworzenia usługi internetowej. Niezbędne dane do eksperymentu zostały wyodrębnione i przekształcone w wygodny do użycia format po szczegółowym przeglądzie tych zbiorów danych. Zaprojektowano i zaimplementowano algorytm treningowy dla 6 modeli, a po ich wytrenowaniu uzyskano wartościowe metryki. Dodatkowo opracowano interfejs po stronie serwera i interfejs sieciowy do interakcji z nimi.

Słowa kluczowe: analiza tekstu, język naturalny, technologie przetwarzania języka naturalnego, NLP, model

Tetiana A. Vakaliuk, Oleksii V. Chyzhmotria, Svitlana O. Didkivska, Illia Linevych

Desarrollo de un servicio web para la creación de pruebas basado en el análisis de textos mediante tecnologías de procesamiento del lenguaje natural

Resumen

El artículo analiza modelos y métodos de procesamiento del lenguaje natural, y selecciona tecnologías modernas para el entrenamiento de modelos con el fin de desarrollar un servicio web para la creación de tests basados en el análisis de textos utilizando tecnologías de procesamiento del lenguaje natural. El estudio considera métodos y algoritmos de análisis inteligente de datos para generar preguntas y respuestas correctas e incorrectas a partir del texto. Los autores justifican la elección de una red neuronal para generar tests basados en textos en inglés y ucraniano y caracterizan las fuentes de datos para el entrenamiento. El estudio también describe la actividad del modelo propuesto, que servirá de base para crear un servicio web. Tras un examen detallado de estos conjuntos de datos, se extrajeron los datos necesarios para el experimento y se transformaron a un formato conveniente para su uso. Se diseñó e implementó el algoritmo de entrenamiento de 6 modelos y se obtuvieron valiosas métricas tras su entrenamiento. Además, se desarrolló una interfaz web y de servidor para interactuar con ellos.

Palabras clave: análisis de texto, lenguaje natural, tecnologías de procesamiento del lenguaje natural, PLN, modelo

Tetiana A. Vakaliuk, Oleksii V. Chyzhmotria, Svitlana O. Didkivska, Illia Linevych

Татьяна Вакалюк, Алексей Чижмотря, Светлана Дидковская, Илья Линевич

Разработка веб-сервиса для создания тестов на основе анализа текста с использованием технологий обработки естественного языка

Аннотация

В статье проанализированы модели и методы обработки естественного языка и выбраны современные технологии обучения моделей с целью разработки веб-сервиса для создания тестов на основе анализа текста с использованием технологий обработки естественного языка. В исследовании используются интеллектуальные методы анализа данных и алгоритмы для генерации вопросов, а также правильных и неправильных ответов из текста. Авторы обосновывают выбор нейронной сети для генерации тестов на основе текста на английском и украинском языках и характеризуют источники данных для обучения. В исследовании также описана работа предложенной модели, которая послужит основой для создания веб-сервиса. Необходимые для эксперимента данные были извлечены и преобразованы в удобный для использования формат после детального рассмотрения этих наборов данных. Был разработан и реализован алгоритм обучения для 6 моделей, после их обучения были получены ценные метрики. Дополнительно были разработаны серверный интерфейс и веб-интерфейс для взаимодействия с ними.

Ключевые слова: анализ текста, естественный язык, технологии обработки естественного языка, НЛП, модель