UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Context-Based Question-Answering System for the Ukrainian Language

Author:
Serhii TIUTIUNNYK

Supervisor:
Vsevolod DYOMKIN

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Serhii TIUTIUNNYK, declare that this thesis titled, "Context-Based Question-Answering System for the Ukrainian Language" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Context-Based Question-Answering System for the Ukrainian Language

by Serhii TIUTIUNNYK

Abstract

This work presents a context-based question answering model for the Ukrainian language based on Wikipedia articles using Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) model, which takes a context (Wikipedia article) and a question to the context. The result of the model is an answer to the question.

The model consists of two parts. The first one is a pre-trained multilingual BERT model, which is trained on the top-100, the most popular languages on Wikipedia articles. The second part is the fine-tuned model, which is trained on the dataset of questions and answers to the Wikipedia articles. The training and validation data is Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016).

There are no question answering datasets for the Ukrainian language. The plan is to build an appropriate dataset with machine translation and use it for the fine-tuning training stage and compare the result with models which were fine-tuned on the other languages. The next experiment is to train a model on the Slavic language datasets before fine-tuning on the Ukrainian language and compare the results.

Acknowledgements

First of all, I thank my supervisor Vsevolod Dyomkin (Ukrainian Catholic University), who provided a lot of useful suggestions.

I thank Artem Chernodub (Ukrainian Catholic University, Grammarly), who mentored our group of students who worked on NLP tasks and helped us to go through the project sprints, asked a lot of reasonable questions.

Also, I would like to thank Grammarly for helping us with computational resources. It helped me to carry out a lot more experiments and achieve better results.

I am grateful to Ukrainian Catholic University and Oleksii Molchanovskyi for managing the master program, which helped me to reboot my career and start the new page of my life.

Finally, I want to thank my groupmates for an unforgettable one and a half year spent together.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
BERT	**B**idirectional **E**ncoder **R**epresentations (from) **T**ransformers
SQuAD	**S**tanford **Q**uestion **A**nswering **D**ataset
SDSJ	**S**berbank **D**ata **S**cience **J**ourney
TF	**T**erm **F**requency
IDF	**I**nverse **D**ocument **F**requency
LSTM	**L**ong **S**ort-**T**erm **M**emory
BAF	**B**idirectional **A**ttention **F**low
GPT	**G**enerative **P**re-trained **T**ransformer
MLM	**M**asked **L**anguage **M**odeling
NSP	**N**ext **S**entence **P**rediction

Dedicated to my family who supported me while studying.

Chapter 1

Introduction

1.1 Problem importance

Nowadays, it becomes more challenging to stay in the context of an expert area without handling large amounts of data. Textual information grows exponentially together with video, audio, photo, and other types of data. Therefore, a model that answers a question is significant. One can use for building chat-bots, automatic quiz generation. Finally, it helps to handle text documents and retrieve the necessary information much faster. For example, it is useful for companies with a massive base of inner instructions. Employees can retrieve the required data based on the scope of the documents.

Along with it, the question-answering system might be beneficial for layers, medical workers, and other specific professions.

Finally, a high performed question-answering system can save much money for call-centers replacing employees working together with text to speech systems.

1.2 General formulation of the problem

The question-answering task is one of the classical problems in natural language processing (NLP). In the standard formulation, the input for the content-based question-answering model has a context and a question to the context. As a context, we can take an article, a document, an essay, a paper, or any other piece of textual information. In this project, we will use paragraphs of articles from Wikipedia. A question is a natural human language question (a query to the system). In this case, articles and questions are in the Ukrainian language. The result of the model is a phrase from the context which contains the answer to the question.

1.3 Goals

This master project was made to enrich NLP tools for the Ukrainian language, in particular, to build a question-answering system for the Ukrainian language. There are the goals for this master project:

1. Adapt a question-answering dataset for the Ukrainian language based on the well-known dataset for the English language like SQuAD (Rajpurkar et al., 2016).

2. Train one of the state-of-the-art models on the Ukrainian language dataset and achieve results near 60-70% according to the SQuAD evaluation.

3. Experimentally explore different pathways of training the models. For example, check if pre-training on the other language datasets improves results for the Ukrainian language.

1.4 Challenges and restrictions

- First of all, there is no question-answering dataset for the Ukrainian language.
- Training state-of-the-art NLP models from scratch is a costly process. For example, training Generative Pre-trained Transformer 2 model (GPT-2) (Radford et al., 2019) proximately costs $40,000 on the Google Cloud Platform, which means training from scratch is not possible in this project.

Chapter 2

Review of related work

Despite the importance of the problem, it is not appropriately solved for the Ukrainian language yet. There was no public result for the Ukrainian language found except some multilingual models like BERT (Devlin et al., 2018), which are not fine-tuned because there are no corresponding datasets for this in the Ukrainian language.

2.1 Classical methods

Let us start a review of existing methods from the classical approaches. Under the term `classical`, we mean methods that use well-known strategies without artificial neural network models. There are unsupervised and supervised methods. Unsupervised approaches are based on word embedding (Al-Rfou, Perozzi, and Skiena, 2013) distances and word frequencies. Supervised methods use labeled dataset for training (logistic regression, support vector machine, etc.). Also, we can attribute logic-based methods (for example, Machine Comprehension Using Commonsense Knowledge (Ostermann et al., 2018)) to the set of classical methods. Such methods are used to solve the question-answering task because logical representations show more abstract concepts, like temporal or logical relation. It is handy for learning a type of commonsense knowledge.

2.1.1 Unsupervised methods

Euclidean distance between sentences The first traditional method we came across during reviewing related works is finding the minimal Euclidean distance between questions and sentences from the context (Swalin, 2018). The idea of this approach is to find an average vector of words for each sentence. The answer to the question is the closest sentence from the context to the question, according to Euclidean distance. It is possible to specify the answer by splitting the sentence into phrases, but it is an additional task that will decrease the accuracy of the method. One more drawback of the described method is relying on the quality of word embeddings. Also, this method does not take into account a dependency between the words in the sentence.

Word and phrase frequency It is possible to use the n-gram approach (Majumder and Mitra, 2002) for generating an answer. The question is parsed into the dependency tree and rebuilt into a narrative sentence with missing the target word or phrase. The missed phrase is filling by the n-gram model. An artificial neural network model can replace the n-gram model. It will be discussed below. The drawback of this approach is the low accuracy of dependency parser models and relying on the phrase frequency in a relatively small volume of text.

2.1.2 Supervised methods

Logistic regression and Support vector machine Supervised traditional methods are also described by Swalin, 2018. The author uses the SQuAD dataset mentioned above for learning. The context is split into the sentences and added to a binary vector. The target sentence is marked as `1`, and all other items are `0`. After that, multinomial logistic regression (Sperandei, 2014) is being trained by the labeled data or support vector machine (SVM) (Kwok, 1998). One of the advantages of this approach is the ability to add some features to the model (dependency between the words, term frequency (TF), inverse document frequency (IDF) (Salton and McGill, 1986), etc.). A term frequency is a feature that increases the weight of frequent words, and inverse document frequency vise verse decreases the weight of widespread words.

2.1.3 Pros and Cons

The advantages of the classical approaches are simplicity and high transparency of the models. Along with it, the model performance on artificial samples is not good enough (near 70% accuracy on the SQuAD validation set). The result will be worse with increasing the size of the context or setting a goal to retrieve more specific answer (a phrase instead of a sentence). Moreover, the results for the Ukrainian language are even worse than the English language. It happens due to higher grammar complexity of the Ukrainian language, fewer text corpora, the presence of word cases, and other language specifics.

2.2 Artificial neural network models

In this part, we will review supervised and unsupervised cases for each main model.

2.2.1 Long short-term memory model

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a recurrent neural network architecture that allows building sequence to sequence models. Also, the input and output vector sizes are not fixed. As an input, LSTM model takes a context and a question and returns a word scores from the context. To connect a vector for context and a vector for a question, we add an attention layer. It is a crucial part of the question answering system based on the LSTM model. The attention layer is a dot product of context and question output vectors. After that, the result of the dot product converts into the probability of being an answer to the question. The approach mentioned above is described in the paper dedicated to Bidirectional Attention Flow (BAF) (Seo et al., 2016).

2.2.2 Generative pre-trained transformer

Generative pre-trained transformer (GPT). There is a second version of this model called GPT-2 (Radford et al., 2019). GPT-2 is one of the state-of-the-art models in language modeling tasks. This model was trained on the Wikipedia articles and internet pages to make the style of generated text more various. This model can only generate the next word based on the previous text. So, to make it answer the question, we have to rephrase questions sentence into a narrative sentence with a skipped phrase for the answer. GPT-2 will generate the answer. The peculiarity of this model is the absence of the context. On the one hand, It can be an advantage if

Model	English	Chinese	Spanish	German	Arabic	Urdu
BERT - Train Cased	**81.9**	**76.6**	**77.8**	**75.9**	**70.7**	61.6
BERT - Train Uncased	81.4	74.2	77.3	75.2	70.5	61.7
BERT - Test Uncased	81.4	70.1	74.9	74.4	70.4	**62.1**
BERT - Z. Shot Uncased	81.4	63.8	74.3	70.5	62.1	58.3

TABLE 2.1: BERT multilingual model performance.

there is no specific data to retrieve the answer. On the other hand, the accuracy will be low for the tasks from special areas (law, medicine, etc.), as the model was not trained on data from the corresponded areas. Anyway, GPT-2 can not be applied to the Ukrainian language, as it is trained only on English texts. Along with it, training the model from scratch or even pre-training on Ukrainian corpora requires a lot of resources and time.

2.3 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) is a transformer-based neural which shows state-of-the-art results in a wide variety of NLP tasks provided by Google researchers. The multilingual BERT model was built for top-100, the most popular languages in Wikipedia, and can be used for a hundred languages out-of-the-box. BERT model training process consists of two stages. The first one is pre-training on the text corpora for a language modeling task. And the second stage is fine-tuning on the question-answer datasets. The first part also requires a lot of resources. As for the fine-tuning, it can be performed even on a single graphics processing unit (GPU).

2.3.1 Multilingual BERT results

There are several modifications of BERT multilingual models that differ by the fine-tuning process. There are BERT fine-tuned on a translated dataset, original dataset (English), cased (use original word case), and uncased (all words are lowercased). Table 2.1 shows the result of BERT modifications on the Cross-lingual Natural Language Inference (XNLI) (Facebook, 2019) dataset (translated datasets). It is a dataset for sentence classification in 15 languages.

As we can see, the best result for the vast majority of languages is provided by model pre-trained on the translated cased dataset. One more important thing is that the cased BERT model fine-tuned on the translated data performs better for non-Latin alphabets languages. Based on the data provided in the Table 2.1, we can conclude that fine-tuning multilingual BERT model on machine-translated datasets shows good enough results on the sentence classification dataset, including non-Latin languages.

By the way, the fine-tuned BERT base model on the English language dataset gives near 37% exact answers on the SQuAD dataset. It can be taken as the baseline result for the context-based question-answering models before this project.

2.3.2 BERT results for the Russian language

The closest existed results to the Ukrainian language are the results for the Russian language (DeepPavlov, 2018), which was gained on the Sberbank Data Science Journey (SDSJ) dataset 2017 (Sberbank, 2017) with a multilingual cased BERT base

model. This model gives 64% of exact matches, and the F1 score is equal to 83% on the SDSJ dataset. This result and results for the XNLI task are pretty good reasons to apply the multilingual BERT model to build a question-answering system for the Ukrainian language.

Chapter 3

Problems and approaches

3.1 Problems

The main objective of this project is to build a question-answer model for the Ukrainian language, which shows accuracy near results shown in Table 2.1 on the well-known benchmarks (SQuAD). The first hypothesis says it is possible to achieve an efficiency near 70-80%, which is close to results for the other languages provided by Google researchers.

Of course, the first target is impossible without an appropriate question-answering dataset for the Ukrainian language. That is why adapting a question-answering dataset for the Ukrainian language based on the SQuAD dataset is a second goal for this project.

One more goal is to compare different approaches for pre-training. Some datasets have human translated data into the Russian and other Slavic languages. It seems that the fine-tuning model on Slavic language datasets and then fine-tuning on the turned into the Ukrainian language dataset might improve performance for the Ukrainian language comparing with direct fine-tuning on the Ukrainian language dataset. So, the next task of this project is to confirm or deny this hypothesis.

3.1.1 Translation problems

To achieve the project goals mentioned above, we need to find an appropriate machine translator to create the dataset in the Ukrainian language, build different model pipelines, and compare results. Also, it might require a human translated small dataset in the Ukrainian language to verify the models.

3.1.2 Articles retrieval problems

Besides, the project needs to retrieve Wikipedia articles in the Ukrainian language. There are articles in the datasets which exist in the English Wikipedia and are absent in the Ukrainian part. So, we have to detect such items and exclude them from the datasets. Moreover, Wikipedia provides articles in the Extensible Markup Language (XML) format, which must be converted into the human-readable text.

The alternative approach for retrieving Wikipedia articles is a direct translation from the existing datasets.

3.2 Approaches

3.2.1 Dataset generation

The very first task is to generate a Ukrainian language dataset from existed datasets (SQuAD) by machine translator. The is a subtask related to machine translation. It is a comparison and checking the quality of the translation. The quality of the translated dataset directly affects the quality of the model. Translation quality can be checked by reverse translation. If the difference between the original text and the text after the forward and the backward translation is small enough, it says about the high quality of translator.

We used Google Cloud API translator to translate the SQuAD into the Ukrainian language. There are many restrictions while using this translator. First of all, translation full SQuAD dataset costs near $450. So checking the quality of translation is a costly process, and we decided not to perform it. The main reason for not doing it has no alternative machine translator to compare results. A second restriction is a number of characters which service allows us to translate per minute. It is equal to 100,000 characters per 100 seconds. The translation of the whole dataset was provided partially applying timeouts to meet restrictions.

3.2.2 Data storing

Generated datasets and articles from Ukrainian Wikipedia are stored in the database to make access to the data more convenient. The translation process required reading and writing data partially. Along with it, the data required additional processing. All these things are more convenient to do while having direct access to records. That is why we decided to store data into a PostgreSQL database system.

There was projected a database structure and built scripts that write the original dataset into the database and reads data from the database to generate the Ukrainian language question-answering dataset in the SQuAD format. Along with it, we built scripts for working with Goggle Cloud translation API and writing translated data into the database.

3.2.3 Data processing

One more problem appeared after the translation step. It is a discrepancy between a translated answer (phrase) without a context and the answer in the context (paragraph of the Wikipedia article). It happens in the Ukrainian language because the same phrase in the context differs from the phrase without the context. For example, it happens due to the declension of words in the Ukrainian language. It is a problem because we have to label a position of the answer in the context. If the translations differ, labeling of the answer position becomes much tricky.

There are two stages for the solution to this problem. The first one is comparing Ukrainian and English language answers. For example, there are pretty many cases when the answer is the year or the other answer in a number format. In this case, we replace the translated answer by the English version and set an answer position in the translated context. This approach covered near 40% of the whole dataset answer positions, but the vast majority of answers are numbers or short answers. It means that this part of the dataset would be biased and does not represent the whole data. That is why we go further and start looking at approaches on how to increase the part of labeled answers.

On the second stage, tokenization and lemmatization were applied to compare answers and context. The `pymorphy2` (Korobov, 2015) and `tokenaize_uk` (Dyomkin and Chaplinsky, 2017) libraries were applied to split context and answers into tokens (words and punctuation characters). After that, `BECYM` (Rysin and Starko, 2019) was applied to each token of the context and answer to lemmatize it. There are two lists (context and answer) of lemmatized tokens, and it is possible to find a correspondence between an answer and a context. In most cases, an algorithm finds correspondences between one or two words, but usually, it is enough to detect a whole lemmatized answer in the context. Sometimes, there are cases when two or more lemmas in the context are corresponding to the answer. It happens when the answer is repeated several times in the context. In such cases, we applied a heuristic method. We find a start position for each candidate, normalize it dividing by the length of the context, and compare it with the same value for the English version of the current instance. The idea is that the translated answer position in the translated context is closer to the original answer position in the original context. We apply normalization to calculate position relative to the length of the context. Then the nearest answer position is taken as an answer. But on the next step, we should join a corresponded number of tokens, and we will get a true answer from the context. Applying the method described above, we managed to increase a part of labeled answer positions up to 88%, which is enough to provide a fine-tuning process and compare our results with the results for the English language dataset.

3.2.4 Models

BERT base model

In this project, we use Bidirectional Encoder Representations from Transformers (BERT) model. Let us dwell upon on the BERT model, and it's pre-training process.

According to the paper Devlin et al., 2018 dedicated to the BERT model, the pre-training process consists of two parts, which force the model to understand languages. The first pre-training task is provided to open the ability of the model to use a bidirectional part. They called this procedure masked language modeling (MLM). The idea is to mask a part of words (near 15%) by a special token `[MASK]` and force model to predict a missed word.

The second pre-training task is the next sentence prediction (NSP). The NSP pre-training process has a huge improvement in the question-answering task as it is based on understanding the relationship between sentences. The dataset is generated from the text corpus. Half of them are true next sentences, and the other half are randomly chosen sentences. The model is forced to detect if it is the true next sentence. That is all for the pre-training process. This part of training is the most costly and can not be carried out as part of this project. That is why we take a pre-trained BERT model.

The most appropriate version of the BERT model (according to the author's recommendation Devlin, 2019) is a multilingual BERT model. The author recommends using the multilingual cased (using an original word case) BERT base model, especially on languages with non-Latin alphabets. We use the BERT-base version, which has 12 layers, 768 hidden layers, 12 heads, and 110M parameters, because the large version can not be used due to computational power restriction, specifically GPU RAM. For example, RTX 2080 ti with 11GB of RAM is not enough for fine-tuning BERT large model even with a batch size equal to one. BERT GitHub documentation

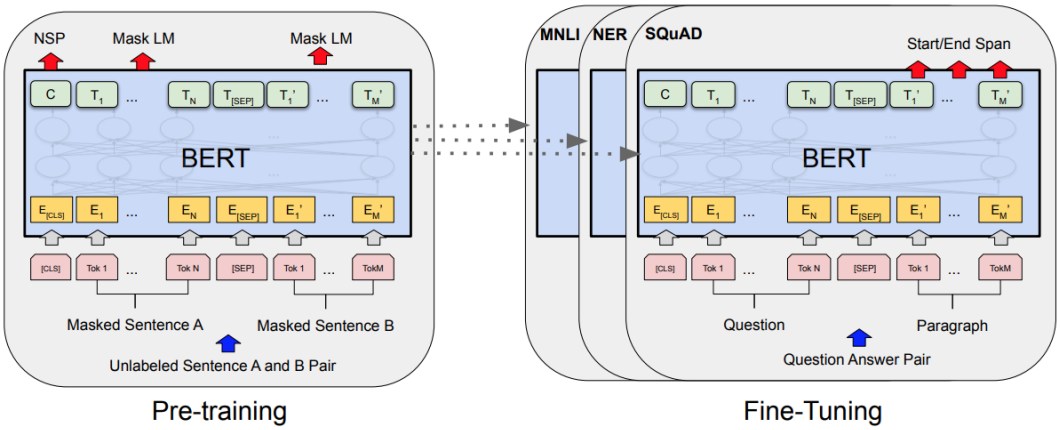

FIGURE 3.1: Illustration of pre-training and fine-tuning processes for BERT (from Devlin et al., 2018).

Devlin, 2019 says that training large models is not possible on a single GPU with 12GB - 16GB of RAM.

So, we have a pre-trained BERT model and the Ukrainian language dataset for fine-tuning. We should mention that the dataset is SQuAD 2.0, which means allowing for the possibility that there is no answer in the context. This fact makes the problem more realistic. The input sequence of tokens consists of the context and the question tokens, which are split by the special token `[CLS]`. The whole dataset is labeled as a position of the start and end token, which corresponds to the answer in the context. If there is no answer in the context, positions of the start and end tokens are the same and equal to `[CLS]` token. The output of the model is probabilities for tokens to be the start and the end of the answer.

$$S_{logit} = S \cdot O_i,$$

$$E_{logit} = E \cdot O_i,$$

where $S \in \mathbb{R}^H$ is a start answer vector, and $E \in \mathbb{R}^H$ is end answer vector, H denotes a dimension of a hidden vector (768 for the BERT base models), O_i - final output of the hidden vector for token i^{th}. Then start (P_i^s) and end (P_i^e) positions equal to:

$$P_i^s = softmax(S_{logit}),$$

$$P_i^e = softmax(E_{logit}).$$

The whole process described above is illustrated in Fig. 3.1[1].

[1] https://towardsdatascience.com/distilling-bert-models-with-spacy-277c7edc426c

Chapter 4

Datasets and Experiments

4.1 Datasets

The question-answering dataset for the Ukrainian language was built from two datasets, specifically from the SQuAD 2.0 Rajpurkar et al., 2016 and SDSJ 2017 Sberbank, 2017. The SDSJ 2017 dataset is built in the format of the SQuAD 1.1, i.e., the answer to the question always exists in the context.

4.1.1 Stanford Question Answering Dataset 2.0

The original SQuAD dataset has the next properties:

- 477 - total number of articles, 442 and 35 - train and test sets respectively;
- 20 239 - total number of paragraphs, 19 035 and 1204 - train and test sets respectively;
- 42 - the average number of paragraphs per article;
- 142 192 - total number of questions, 130 319 and 11 873 - train and test sets respectively;
- $\frac{2}{3}$ part of questions have an answer in the context, and $\frac{1}{3}$ of questions are marked as impossible to answer.

The human performance for the original SQuAD 2.0, according to the *The Stanford Question Answering Dataset rating*, is 86.8% of exact matches, and the F1 score is 89.5%. The modern models give results higher than human performance.

After the translation of the original dataset into the Ukrainian language, we managed to label near 88% of the original answers. It covered each paragraph from the train and test sets.

Translation SQuAD 2.0 dataset into the Ukrainian language costs near $450 in the Google Cloud Translation API Platform.

4.1.2 Sberbank Data Science Journey 2017 dataset

This dataset was prepared by Sberbank for the competition in 2017. SDSJ 2017 is built in the SQuAD 1.1 format (only existing answers). The peculiarity of the dataset is that it has one question per paragraph. Along with it, each paragraph is related to a separate article. Paragraphs are not joined by an article.

The SDSJ 2017 dataset has the next properties:

- 50 364 - number of paragraphs and questions, 45 328 and 5036 - train and test sets, respectively;

- 38 779 (77% of the original dataset) - number of paragraphs and questions with labeled answer position in the context, 34 910 and 3869 - train and test sets respectively.

Translation SDSJ 2017 dataset into the Ukrainian language costs near $610 on the Google Cloud Translation API Platform.

There is a public result for the SDSJ 2017 dataset using the cased multilingual BERT base model published by DeepPavlov, 2018, and it is equal to 64% of exact matches, and the F1 score is equal to 83%.

4.2 Experiments

First of all, we would like to mention that we use for our experiments an open-source adaptation of BERT in Tensorflow (Devlin, 2019) to a question-answering task in the SQuAD format. The fine-tuning process was carried out on a single GPU, RTX 2080 Ti (11GB of RAM) with TensorFlow 1.14.0, and a single epoch usually takes 3-4 hours. All results are provided according to the SQuAD 2.0 benchmark (Rajpurkar et al., 2016). One can take a script for model validation on the *The Stanford Question Answering Dataset rating*. The evaluation method returns two metrics. The first one is the exact matches (EM):

$$EM = \frac{\sum_{i=1}^{N} I(x_i)}{N},$$

where

$$I(x_i) = \begin{cases} 1, & \text{if } x_i \text{ exactly equals to the original answer,} \\ 0, & \text{otherwise} \end{cases}.$$

As there are impossible answers in the SQuAD 2.0 dataset, evaluation script also returns F1 score:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall},$$

where

$$precision = \frac{EM}{EM + \text{false positive}},$$

$$recall = \frac{EM}{EM + \text{false negative}}.$$

4.2.1 Separated datasets

One of the first experiments was carried out on the original SDSJ 2017 dataset, and the result published by DeepPavlov, 2018 was almost repeated. After eight epochs of fine-tuning with a start learning rate equal 2×10^{-5}, we got 62% of exact matches, and the F1 score is 81% (64% and 83% are the published results respectively). So we almost repeated an achieved result without tuning hyperparameters. After that, there was provided an experiment for the translated SDSJ 2017 dataset with the same hyperparameters. The result is 63.3% and 81.4%, respectively, which is very close to the results for the original dataset.

The next experiments were carried out on the translated SQuAD dataset. The hyperparameters are learning rate: 3×10^{-5}, batch size: 2, max sequence length: 384 tokens, and the number of epochs: 5. The result is 54.6% and 60.5%. By the way,

Sequence length	Max batch size
120	24
180	16
228	12
256	8
384	6
512	4

TABLE 4.1: Maximum batch size on single RTX 2080 ti GPU (11GB RAM) with TensorFlow 1.14.0

increasing the number of training epochs to 10 did not improve the result. It became even a little bit worse.

Increasing batch size to 4 by increasing computing resources improved the previous result a little bit (56.1% and 62.2%).

Then it was time to check the hypothesis described in the list of goals. We checked if a fine-tuning on the translated SQuAD datasets, which was fine-tuned on the original SDSJ 2017 dataset, improve the result for the SQuAD dev dataset. The training processes lasted 5 epochs for each dataset with learning rate: 3×10^{-10}, max sequence length: 384, batch size: 4. The hypothesis was not confirmed, and the result was almost the same (56% and 62%) as fine-tuning on the SQuAD from scratch (56.1% and 62.2%).

4.2.2 Joined datasets

When our hypothesis was not confirmed, we decided to join translated SQuAD and translated SDSJ datasets and continue experiments with it. The first experiment with the joined datasets lasted 5 epochs, learning rate: 2×10^{-10}, batch size: 4, and max sequence length: 384. It improved the result a little bit. The exact match score became 58.1%, and the F1 score is 63.9%.

After that, we tried to transliterate the Ukrainian language dataset and apply a fine-tuned BERT model for the English language. Fine-tuned BERT base model on the original SQuAD 2.0 showed pretty good results on the Ukrainian language dataset from scratch (37% and 42%). After fine-tuning, it became 49.2% and 53.7%, which is less than the multilingual model result. The fine-tuned on the translated data multilingual model gives 57.7% of exact matches, and the F1 score is 63.2%.

The next set of experiments was dedicated to the combination of max sequence length and batch size. There is a restriction on the max sequence length and batch size. There is a table of restrictions for single Titan X GPU (12GB RAM) with TensorFlow 1.11.0 in the Google research GitHub BERT documentation (Devlin, 2019), which says that max batch size value for the sequence length 384 is 12. We are using RTX 2080 ti GPU (11GB RAM) with TensorFlow 1.14.0, and the experiments show that batch size can not be more than 6 for the sequence length 384. More restrictions for our configuration are provided in Table 4.1. It occurs that only 0.6% of the translated SQuAD examples have token sequence length more than 256. For the max sequence length equal to 256, there is an ability to use batch size equal 8. Increasing a batch size to 8 and a learning rate to 5×10^{-5} allow multilingual BERT model to give 59.1% of exact matches, and the F1 score 64.3% after 5 epochs of fine-tuning. The best result we get is 61.2% of EN, and the F1 score is 65.6%, which was gained after 5 epochs, learning rate: 5×10^{-5}, batch size: 16, and sequence length: 180.

Model	Dataset	Seq. len.	BS	LR	Epoch	EM	F1
Multiling. cased BERT-base	SQuAD uk	384	2	3e-5	5	54.6	60.4
Multiling. cased BERT-base	SQuAD uk	384	2	3e-5	10	54.2	60.3
Multiling. cased BERT-base	SQuAD uk	384	4	3e-5	8	56.1	62.2
Multiling. cased BERT-base	SDSJ ru	384	4	2e-5	8	61.8	81.2
Multiling. cased BERT-base	SDSJ uk	384	4	2e-5	10	63.3	81.4
Multiling. cased BERT-base	SDSJ ru -> SQuAD uk	384	4	3e-5	5+5	56.02	62.03
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	384	4	2e-5	5	57.7	63.7
Multiling. cased BERT-base	SQuAD uk + SDSJ uk translit.	384	4	2e-5	5	57.7	63.2
Cased BERT-base	SQuAD uk + SDSJ uk translit.	384	4	2e-5	5	49.2	53.7
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	384	4	2e-5	5	57.7	63.2
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	512	4	5e-5	2	57.8	63.2
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	256	8	5e-5	5	59.1	64.3
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	120	24	5e-5	5	59.5	63.9
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	228	12	5e-5	5	60.2	65.5
Multiling. cased BERT-base	SQuAD uk + SDSJ uk	180	16	5e-5	5	61.2	65.6

TABLE 4.2: Experiment results. Seq. len. - max token sequence length; BS - batch size; LR - learning rate; EM - exact matches.

All experiment results are provided in Table 4.2. As we can see, sequential fine-tuning on the Russian dataset and then on the Ukrainian dataset did not give an improvement. But the joining translated SDSJ and SQuAD datasets gave a little bit better results. Along with it, increasing a batch size at the expense of sequence length decreasing also showed a small improvement. In our case, the optimal batch size is close to 16, which allows applying sequence length equal to 180. From which we can gather that increasing batch size having a pretty large sequence length might improve the performance.

61% of exact matches by the SQuAD benchmark looks like a good performance, but we decided to calculate another score. What if the predicted answer is in the labeled answer or vice versa. So, the answer does not match the labeled value, but it is a part of the etalon answer. We decided to count predicted answers, which are longer than etalon answers, and etalon answer is more than 30% of the predicted answer. In this case, a precision equals 66.6%.

Also, we decided to check the performance of the best our model by SQuAD benchmark on the Ukrainian language dataset (SQuAD + SDSJ) on the original Russian language dataset (original SDSJ). The result (63.8% of exact matches and F1 score 82.0%) was a little bit better than the model fine-tuned only on the original data (61.8% and 81.2%, respectively). It can be explained by the bigger training set, which consists of both training sets.

Finally, we built a small Ukrainian language dataset (13 contexts and 100 questions) manually. It was prepared to check if the model was not overfitted on the translated dataset, and the model can be applied to the real word Ukrainian texts. And surprisingly, we got the result 73%, which is better than on the adapted dataset (61.2%). This result can be explained by a higher context and question clarity than the translated texts.

Chapter 5

Conclusions

5.1 Contribution

This project is dedicated to the question-answering system for the Ukrainian language, and according to the goals set at the very beginning, we achieved the following results, which we are contributing:

1. For the first time, a question-answering dataset for the Ukrainian language based on the well-known dataset for the English language (SQuAD 2.0 Rajpurkar et al., 2016) and Russian language (SDSJ 2017 Sberbank, 2017) was adapted[1], which allowed to fine-tune and compare QA models for the Ukrainian language with models for the other languages by the SQuAD[2] benchmark through the use Google Cloud Translation API and applying translated data processing, specifically comparing translated context and answers in lemmatized forms and matching lemmatized tokens. So, we have the adapted version of SQuAD 2.0[3], SDSJ 2017[4], aggregation of SQuAD 2.0 and SDSJ 2017[5], and a manually-created small test dataset[6].

2. For the first time, the question-answering model was fine-tuned for the Ukrainian language[7] on the Ukrainian language question-answering dataset, which shows performance near 61% by the SQuAD benchmark through the use pre-trained state-of-the-art models, like multilingual BERT base model for the Ukrainian language.

3. There were carried out experiments with fine-tuning multilingual BERT model on the original Russian language dataset (SDSJ 2017) then fine-tuned on the adapted Ukrainian language dataset and compared results with model, fine-tuned only on the Ukrainian dataset, which allowed to check if fine-tuning on the similar languages improves the performance for the Ukrainian language by the SQuAD benchmark. It turned out that it does not improve performance. It occurred that combining both datasets improves performance (Table 4.2).

[1] https://github.com/s-e-r-g-y/context-based-qa-for-uk
[2] https://rajpurkar.github.io/SQuAD-explorer/
[3] http://lang.org.ua/static/downloads/squad/squad-2.0-uk.zip
[4] http://lang.org.ua/static/downloads/squad/sdsj-2017-uk.zip
[5] http://lang.org.ua/static/downloads/squad/squad-2.0+SDSJ-uk.zip
[6] http://lang.org.ua/static/downloads/squad/dev-human-v2.0.json
[7] http://lang.org.ua/static/downloads/squad/multi_cased_bert_base_uk.zip

5.2 Future work

The context-based question-answering system is a complex task, especially for the low-resourced languages in terms of NLP solutions. So, we have several ideas for future improvements:

- First of all, it requires a vast number of clean data. So, we can increase the quality of the adapted data by applying auxiliary models and algorithms. For example, approaches for finding correspondence between context and answer tokens. Also, we can try to generate QA datasets from the corpus of the Ukrainian language. It is a very ambitious task.
- One more possible improvement, which is worth to check, is applying the ensemble of the models (Zhou, Zhang, and Jiang, 2019). It can be a BERT base + multilingual BERT base + multilingual BERT large providing GPU/TPU allows using it.
- For some reason, this task can be applied for finding an answer and a context of the answer in the large documents. It is possible to extend BERT models for such that task.
- Finally, it is possible to increase the performance of the original task by just applying more powerful GPU/TPU for fine-tuning or even pre-training BERT models. In this case, we can apply BERT large models instead of the base model.

Bibliography

Al-Rfou, Rami, Bryan Perozzi, and Steven Skiena (2013). *Polyglot: Distributed Word Representations for Multilingual NLP*. arXiv: 1307.1662 [cs.CL].

DeepPavlov (2018). "Question Answering Model for SQuAD dataset". In: URL: http://docs.deeppavlov.ai/en/master/features/models/squad.html.

Devlin, Jacob (2019). *BERT GitHub*. https://github.com/google-research/bert.

Devlin, Jacob et al. (Oct. 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: arXiv: 1810.04805 [cs.CL].

Dyomkin, Vsevolod and Dmitry Chaplinsky (2017). "Tokenize UK". In: URL: https://github.com/lang-uk/tokenize-uk.

Facebook, Research (2019). *XNLI*. https://github.com/facebookresearch/XNLI.

Group, Stanford NLP. *The Stanford Question Answering Dataset rating*. URL: https://rajpurkar.github.io/SQuAD-explorer/.

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Korobov, Mikhail (2015). "Morphological Analyzer and Generator for Russian and Ukrainian Languages". English. In: *Analysis of Images, Social Networks and Texts*. Ed. by Mikhail Yu. Khachay et al. Vol. 542. Communications in Computer and Information Science. Springer International Publishing, pp. 320–332. ISBN: 978-3-319-26122-5. DOI: 10.1007/978-3-319-26123-2_31. URL: http://dx.doi.org/10.1007/978-3-319-26123-2_31.

Kwok, James Tin yau (1998). "Automated Text Categorization Using Support Vector Machine". In: *In Proceedings of the International Conference on Neural Information Processing (ICONIP*, pp. 347–351.

Majumder, Prasenjit and Mandar Mitra (2002). "N-gram: a language independent approach to IR and NLP". In:

Ostermann, Simon et al. (June 2018). "SemEval-2018 Task 11: Machine Comprehension Using Commonsense Knowledge". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 747–757. DOI: 10.18653/v1/S18-1119. URL: https://www.aclweb.org/anthology/S18-1119.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In:

Rajpurkar, Pranav et al. (June 2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: arXiv: 1606.05250 [cs.CL].

Rysin, Andriy and Vasyl Starko (2019). "Large electronic dictionary of the Ukrainian language". In: URL: https://r2u.org.ua/articles/vesum.

Salton, Gerard and Michael J McGill (1986). "Introduction to modern information retrieval". In:

Sberbank (2017). "Sberbank Data Science Journey". In: URL: https://sdsj.sberbank.ai/2017/ru/contest.html.

Seo, Minjoon et al. (2016). *Bidirectional Attention Flow for Machine Comprehension*. arXiv: 1611.01603 [cs.CL].

Sperandei, Sandro (Feb. 2014). "Understanding logistic regression analysis". In: *Biochemia medica* 24, pp. 12–8. DOI: 10.11613/BM.2014.003.

Swalin, Alvira (May 2018). *Building a Question-Answering System from Scratch*. URL: https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507.

Zhou, Wen, Xianzhe Zhang, and Hang Jiang (2019). "Ensemble BERT with Data Augmentation and Linguistic Knowledge on SQuAD 2.0". In: URL: https://pdfs.semanticscholar.org/2f99/5b11ab14637458e7b8c2b592b0ceedc3b875.pdf.