Small Can Be Beautiful in LLMs for SSH: a Case for Bulgarian

Nikolay Paev, Kiril Simov, Petya Osenova, Teodor Valchev, Stefan Marinov
Artificial Intelligence and Language Technology
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Bulgaria
{nikolay.paev, stefan.marinov}@iict.bas.bg
{kivs, petya}@bultreebank.org
teodorvulchev@gmail.com

Abstract

In the paper we present a set of small LLM-based models for solving basic NLP tasks for Bulgarian — POS tagging, Lemmatization, Dependency parsing, Named Entity Recognition, Named Entity Linking, Event Annotation, among others. In order to create fine-tuned models for these tasks, we first pre-train models using architectures like BERT, Modern-BERT, and T5 with different sizes, over Bulgarian data only. For each of the tasks we report our approach towards the fine-tuning, the results from the experiments and also the evaluation. Then we define a way to visualize the results over HTML documents which contain the analyzed texts. Our rationale is as follows: most, if not all SSH research scenarios, need a reliable processing chains that can be customized with respect to the specific needs. These scenarios would also need proper visualization for human observation. We aim to provide such a basic LLM-based toolkit.

Keywords: Bulgarian pre-trained LLMs, LLM-based NLP models for Bulgarian, Visualization

1. Introduction

Although it is widely accepted that the power of the Large Language Models[1] (LLMs) is in their ability to solve more complex tasks such as *Question Answering*, *Summarization*, **Information Retrieval**, **Chatbots**, and many more, we aim at creating a set of LLM-based models for the basic NLP tasks in Bulgarian. These are *Tokenization*, *Part-of-speech tagging*, *Lemmatization*, *Parsing (constituent or dependency syntax)*, *Named Entities Recognition*, *Named Entities Linking*, *Word Sense Disambiguation*, *Event recognition*, *Co-reference Resolution*, *Textual Entailment*, *Sentiment Analysis*, and others. In this paper, we present models for most of the above tasks, except for the last three ones, on which we are working at the moment.

In our opinion, addressing even the more basic tasks[2] can be very useful to support research activities in areas like Social Sciences, Humanities, Linguistics. In addition, such tasks might help to solve more complex problems in these areas. The aforementioned tasks (as we demonstrate in this paper) could be solved by relatively small LLMs such as BERT, T5, and similar. In this way, many more experiments can be planned and conducted, thus opening opportunities to test different approaches and achieve better results. Here we present the state-of-the-art for Bulgarian for each of the following tasks: *Tokenization*, *Part-of-speech tagging*, *Lemmatization*, *Parsing (constituent or dependency syntax)*, *Named Entities Recognition*, *Named Entities Linking*, *Word Sense Disambiguation*, *Event recognition*.

Each of the basic tasks requires to be fine-tuned on a set of appropriate data. For Bulgarian we rely on datasets that are freely available such as **BulTreeBank**, **Bulgarian Event Corpus**, **The Bulgarian part of the Balto-Slavic Corpus**. In addition, we annotated some extensions of the aforementioned or new datasets to support and improve the models.

We are aware that LLMs with billions of parameters can also be used to solve these tasks. But

[1] We are aware of the fact that there is a tendency for assuming that LLMs have more than ten billion parameters. In this paper we adhere to a broader definition and we consider an LLM each language model based on a Transformer architecture. Thus, we work with transformer models with less than two billion parameters. Here we refrain from using the term *small language models*, because a new trend emerged in the development of the field, where this term is used for another type of transformer models — reduced versions of large language models with many more than 10B parameters. They are results from applying methods like *https://github.com/jamwithai/production-agentic-rag-course*, *Pruning*, and *Quantization* the more than 10B parameters models.

[2] Another classification of tasks in NLP is intrinsic vs. extrinsic tasks which are usually used in the evaluation of different NLP applications. The nowadays LLM approaches to NLP assume that intrinsic tasks are no longer necessary, because the users need output from the extrinsic category (Question Answering, Dialog Systems, Information Extraction, Summarization, .etc)

Proceedings of Shaping Multilingual, Multimodal AI for the Social Sciences and Humanities (LLMs4SSH) @ LREC 2026, pages 187–197
11 May 2026.

they are not ready off-the-shelf to do this as many of the users expect. We performed some experiments with ChatGPT 4.0 services to implement these tasks using a prompting approach. For tasks like POS, Lemmatization, Universal Dependencies (UD) Parsing, the results were worse than ours.[3] For the WSD task, both approaches perform similarly. We did not conduct experiments based on fine-tuning. We imagine that such fine-tuned models will be comparable. However, generally this means that the creation of manually annotated data is still a central task for such models.

The main contributions of this work include: (1) Pre-training of LLMs with a small number of parameters used for the fine-tuning of the NLP tasks. The good results demonstrate that such models are sufficiently effective for the tasks. (2) We implemented a set of fine-tuned models for Bulgarian NLP tasks, which improved the results of the models previously known to us. Some of the models are the first of their kind to be trained on only-Bulgarian data. (3) We created a scheme that enables the integration of all annotations and their visualization in a user interface.

The structure of the paper is as follows: in the next section we present some related works. In Sect. 3 we introduce the main language resources that are used in the fine-tuning of the models. In Sect. 4 all the models that we pre-trained and fine-tuned for the different tasks are presented. Sect. 5 describes how we incorporate the annotations produced by the different models within an HTML editor in order for the users to have the possibility to edit the documents and to have access to the annotations. Such possibilities will be very useful for interested colleagues in the area of Humanities. The last section concludes the paper.

Some citations in the paper are temporarily excluded in order to meet the anonymity requirement.

[3]For example, we tried asking ChatGPT about the morpho-syntactic features of the marked phrase in the sentence:
(**BG**) Struvalo mu se sramno da razkrie, che `e roden` brat na brodyaga s lice na prestypnik.
(**EN**) He felt ashamed to reveal that he `was the biological` brother of this vagrant with the face of a criminal.
The phrase "`was the biological''` in Bulgarian is homonymic to "`was born`". But in the example the real use of the form "roden/ADJ" (biological) is an adjective. However ChatGPT categorized it as a participle which would be correct in the case of "was born". We tried with several different prompts, but the result was the same. Our fine-tuned classification model predicts the correct use. Similar for other examples and tasks.

2. Related Work

For each of the tasks, we present here some related work. The LLM approaches to dependency parsing rely on some observations over the learning of syntactic information by the models. For example Zhou et al. (2023) show that prepositional phrase attachment poses the biggest challenge to understanding syntax by LLMs. The case study on the training dynamics of LLMs revealed that most syntactic knowledge is learned during the initial stages of training. In some cases, syntactic knowledge is encoded directly in the LLM. For example, Shen et al. (2021) propose a new syntax-aware language model — Syntactic Ordered Memory (SOM). The model explicitly models the structure with an incremental parser and maintains the conditional probability setting of a standard language model (left-to-right). In our case, instead of an incremental approach, we use predefined partial syntactic information. With respect to dependency parsing Özates et al. (2020) use special rules to introduce dependency relations between certain word forms in sentences. Each rule identifies some arcs within the dependency tree. In our implementation of dependency parsing, we follow the approach of McDonald et al. (2006) about a graph-based dependency parsing performed in two steps: (1) determination of dependency arcs in the syntactic tree — the immediate domination relation over the tokens in the sentence — for each token to find its immediate parent token (adding special token for the root of the sentence); and (2) labeling the selected arcs with the appropriate dependency relations.

Word sense disambiguation is the task of determining the sense of a polysemic word in a specific context. In our work, we started with a set-up very close to the one described in (Huang et al., 2019). In their work, they construct *context-gloss* pairs. These pairs are constructed by combining the word form for which we want to assign the correct sense in a context (the text in which the word form appears) and the glosses from wordnet. Each of these pairs are classified as correct or not, depending on the gold annotation. In this way, the training examples are constructed. There are some variations in the approach to constructing training examples depending on whether or not the selected word is marked. In some cases, additional knowledge is added to the examples, depending on the context of the senses within WordNet — see (Song et al., 2021b).

The event extraction task is very often defined as consisting of two subtasks: (1) *Event Detection* (ED), and (2) *Event Argument Extraction* (EAE) — see (Simon et al., 2024), (Lai, 2022), and citations within them. In the first task, the system is expected to identify the span and the type of events. Usually,

the event detection starts with a trigger recognition since triggers anchor the events. The second task identifies the arguments (participants) of the event in the text and relates them to their roles. When the two tasks are solved separately by different models, the result is called a *pipeline* approach. In cases where the model solves both tasks together, the result is called a *joint* approach. We implemented a joint generative event extraction system that produces a formal textual representation of the extracted event information. More specifically, we follow the `Text2Event` approach to EE (Lu et al., 2021). `Text2Event` defines an end-to-end generative model that transforms an input of tokens into a linearized event structure.

The paper defines the linearized representation as an S-expression that for each event occurrence contains an expression pair containing the type *the type of the event* and *the corresponding text span* followed by a list of *role* and *span* pairs representing the arguments of the event mentioned in the text. The transformed dataset (for example, ACE, mentioned above) has then been used to train a `T5` encoder-decoder language model (Raffel et al., 2019b). In order to restrict the output to the required event representation, the authors explore the *constrained decoding* that provides a mechanism for exploiting the knowledge of an event schema to form the output.

For the visualization of the annotations, we rely on our own experience with the following systems: the GATE Teamware — (Bontcheva et al., 2013), the INCEpTION platform — (Klie et al., 2018), SpaCy: Industrial-Strength Natural Language Processing[4]. All of them have functionality for creation of rules for automatic text processing including regular expression rules and programming languages — Java, Python, for processing the predefined document data models. Neves and Ševa (2019) provide a comprehensive review of manual annotation tools. They defined a set of evaluation criteria for what makes an annotation tool useful.

NB: In the last version, we will include more related works.

3. Datasets for Bulgarian

In this section, we present the main datasets used in our work as a basis for fine-tuning of the different NLP models.

- *BulTreeBank*: the original constituent variant (Simov et al., 2002) contained 256 000 tokens. We extended it to the morphological level with 2 445 407 tokens.
- *The Bulgarian part of the Balto-Slavic Corpus*: For the shared task (Piskorski et al., 2021) the corpus has been annotated with Named Entities (NEs) adhering to pre-defined guidelines, and also the NEs were linked cross-lingually through the lemmas. On this available corpus, we mapped the NEs to Wikipedia URLs and processed the texts on the morphological level. In addition, we lemmatized all the tokens. The total amount of tokens is 400 000.
- *The Bulgarian Event Corpus (BEC)*: BEC (Simov et al., 2025) comprises 227 documents with 291 196 tokens of texts that are in the area of history, biographies, ethnography, etc. The corpus has been annotated on two levels: NEs and events with triggers and roles. In addition, the co-reference chains were marked as well.

4. LLM-based NLP Models for Bulgarian

Here we describe the construction of the Bulgarian NLP models based on LLMs. Our understanding is that such models have to be relatively small in order for them to be easily re-trained when necessary. The resulting models are also much easier to deploy, as they have modest hardware requirements. We follow the well established paradigm of pre-training language models on a large collection of text and later fine-tuning them for specific tasks on smaller sized supervised datasets.

4.1. Pre-trained models

In this section, we introduce our pre-training setup and the produced models. For the pre-training, we use unsupervised corpora of 29B Bulgarian tokens. We sourced them from 3 openly available datasets: CulturaX (Nguyen et al., 2024), Macocu (Bañón et al., 2023), and HPLT (de Gibert et al., 2024), as well as from some other smaller datasets that we gathered manually, such as News sites (processed manually (5 BW)), Wikipedia, PHD theses, research papers, and other. All data is publicly available. The dataset was deduplicated on document level with approximated Jaccard Similarity using MinHashLSH from datasketch package.[5] We used a threshold of 0.7. We have pre-trained various types of models — *encoder only, encoder-decoder, decoder only*. They are suitable for different NLP tasks. In the work reported here, we present only the first two types of models. The pre-trained models are publicly available on HuggingFace[6].

[4] https://spacy.io/

[5] https://ekzhu.com/datasketch/

[6] https://huggingface.co/AIaLT-IICT.

4.1.1. Encoder models

We consider the BERT (Devlin et al., 2018) architecture the most popular encoder architecture. We developed BERT models of different sizes. All use a vocabulary of 50 176 tokens processed with the WordPiece tokenization tool — Song et al. (2021a). Recently, a modern and more efficient version of the BERT architecture was proposed in Warner et al. (2024). Following the ModernBERT architecture, we also pre-trained a number of models with various sizes. The basic characteristics of the models are presented in Table 4.1.1. It is well known that such types of model are widely used for classification tasks as well as for producing text embeddings. In our work, we use them for tasks like POS Tagging and Token classification. We also use them for Universal Dependencies oriented parsing.

Model	casing	dim	layers	pars
bert-base	both	768	12	124M
bert-large	both	1024	24	355M
bert-XL	no	1024	48	657M
mbert-base	no	768	22	149M
mbert-large	no	1024	28	395M

Table 1: BERT and ModernBERT based pre-trained encoder models. The first two models are available in both — cased and uncased variants — while the others are uncased. The columns denote: name of the model, casing type, token embedding size (**dim**), number of the transformer layers (**layers**), and number of parameters (**pars**).

The training procedure used the Masked Language modeling objective (as proposed in the BERT paper) with 20% noise tokens. We trained for 3 epochs with a learning rate of 1×10^{-4}, batch size of 256 chunks, and context-length of 512 tokens. For applications requiring longer context we extended the model context-length by a subsequent training on longer sequences (from 4096 to 8192 tokens).

4.1.2. Encoder-Decoder models

As mentioned above, encoder models excel especially in classification tasks. Unfortunately, not all tasks can be modeled in this way. An example for this is the lemmatization task: Each word form must be mapped to its lemma, with the possible lemmas being too many to be efficiently modeled as classes. Thus, a generative model capable of making a good representation of the input text seems to be more suitable. Thus, we considered the architecture of T5 (**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer) (Raffel et al., 2019a), which combines an encoder and a decoder. We trained different sized T5 models, presented in Table 2.

The training procedure used a span denoising objective as proposed in the original paper with a noise density of 25% and a mean noise span of 3 tokens. We used the same hyperparameters for the training as in the encoder models. We also used the SentencePiece BPE tokenizer (Kudo and Richardson, 2018) since it proved to be better suited for text generation.

Furthermore, some sequence-to-sequence applications benefit from lower level tokenization (for example, spellchecking). We trained T5 models on character level tokenization. The training objective was again span denoising with a mean noise span of 7 characters, since 7 letters is the average length of the words in the corpus. Since the character level tokenization segments the training corpus in more tokens, we used a version of the pre-training corpus filtered with a more aggressive deduplication to 10B words.

Arch	tokenization	dim	layers	pars
T5	subword	1024	12-12	403M
T5	character	1024	16-16	470M
T5	subword	1536	16-16	1.1B

Table 2: T5-based pre-trained models. The first and third models used uncased subword level tokenization, while the second model employed a character level tokenization. The first column describes the types of the architecture, the second column shows the tokenization level, the third column (**dim**) presents the token embedding size, the fourth (**layers**) — the number of the transformer layers for each model, and the last column (**pars**) contains the number of parameters for each model.

4.2. Task specific fine-tuning

After pre-training, the models can be tuned for various specific tasks. This subsection describes our efforts with respect to some of them.

4.2.1. Encoder based classification

Many tasks can be modeled as classification — assigning a label from a fixed number of classes to each token in the text or the whole text. We fine-tuned our pre-trained encoder models on different tasks, as well as the publicly available at HuggingFace *bert-web-bg*[7] and *bert-web-bg-cased*[8] which is one of the few Bulgarian Transformer models available. Below are more details about the tasks. Table 3 gives test result reports, comparing the pre-trained models.

[7] https://huggingface.co/usmiva/bert-web-bg

[8] https://huggingface.co/usmiva/bert-web-bg-cased

Pre-trained Model	**Tok+SS** (F1)	**NER** (F1)	**XPOS** (Acc)	**UAS** (Acc)	**Text Err** (F1)	**WSD** (F1)
bert-web-bg	0.9669	0.9198	0.9787	0.9033	0.8511	0.7756
bert-web-bg-cased	0.9849	0.9005	0.9810	0.9192	0.8866	0.7507
bert-base	0.9877	0.9289	0.9865	0.9564	0.9214	0.7970
bert-base-cased	0.9927	**0.9497**	0.9858	0.9488	0.9131	0.7949
bert-large	0.9895	0.9378	**0.9878**	0.9412	0.9225	0.8155
bert-large-cased	**0.9940**	**0.9497**	0.9877	**0.9571**	0.9245	0.8136
modernbert-base	0.9902	0.9174	0.9864	0.9523	**0.9462**	0.8119
modernbert-large	0.9895	0.9206	0.9875	0.9444	0.9423	**0.8244**

Table 3: Test results across encoder fine-tuning tasks. Test results on the classification tasks. Rows show results for the following tasks: Tokenization and Sentence Segmentation, Named Entity Recognition, Part of speech and morphological tagging (**XPOS**), Unlabeled syntax head prediction (Unlabeled attachment score - **UAS**), Text error tagging and Word Sense Disambiguation. Macro F1 is reported for Tokenization+Sentence Segmentation, NER, Text Error tagging, and WSD. Accuracy is reported for the remaining tasks.

Tokenization and Sentence segmentation is a task that classifies the beginning of the words and the sentences. In Bulgarian sentence and word boundaries are ambiguous, and thus a model is needed. [9] Often tokenization and sentence splitting is the first objective to be completed in the NLP pipe, since the other annotations usually work on sentence level. We model the task by assigning to each token a class from B-SEN, B-WOR, and I-WOR. These correspond to the beginning of a first word of a sentence, the beginning of a non-first word of sentence, and the continuation of a word. Any supervised tokenized dataset with consequent sentences can be used as training data. Here, the *Bulgarian Event Corpus* has been considered. We divided the corpus into 766 train, 96 validation, and 96 test chunks of 512 tokens. We train for 20 epochs with a linearly decaying learning rate starting from 1×10^{-4} and batch size of 8*32. We pick the best model by validation loss (usually it is around 17 epoch with negligible increase of the val loss after).

Named Entity Recognition is the classical task of assigning labels to word spans that are classified as names. We use *The Bulgarian part of the Balto-Slavic Corpus*. It contains 11 classes with 5 NER categories: person, location, organization, event, and pronoun. We use the provided train and test splits. We partition a validation split from the train data. That amounts to 6 032 train, 671 validation and 2 180 test sentences. We train for 5 epochs with a linearly decaying learning rate starting from 2×10^{-5} and batch size of 8*16. We pick the best model by validation loss (usually it is around 17 epoch with negligible increase of the val loss after).

Part of speech tagging is the task of assigning a label corresponding to the grammatical features of the word, grouped in POS tags. For training, we use our extended version of the *BulTreeBank* corpus. We split the data into 134 331 train, 7 071 validation, and 15 712 test sentences. We train for 3 epochs with linearly decaying learning rate from 5×10^{-5} and batch size of 8*16 sentences. Our models score around 0.99 accuracy, which to our knowledge is the SOTA for the Bulgarian on this task.

Text error tagging is the task of finding spelling, grammatical, and punctuation errors in a given text. We developed data for this task by noising a text that we consider correct with various rules corresponding to common mistakes, including casing errors. In this way, we automatically created training dataset of 11 394 612 tokens in 188 908 text chunks. We model the task as binary token classification and train for 1 epoch with linearly decaying learning rate from 2×10^{-05} and batch size of 8*16 text chunks. During training of the uncased models, we ignore the casing error labels. For testing, we developed a small set of error correction exercises from publicly available Bulgarian exams consisting of 3 101 tokens in 46 texts. On this set, both cased and uncased models are tested. The Macro F1 test scores are presented in Table 3. Manual evaluation showed that the uncased models spot difficult punctuation errors better than the cased models but are naturally unable to assess casing errors. Thus, we got the best results by combining the error predictions from the best uncased model and the best cased model, trained only on casing errors.

4.2.2. UD syntax parsing

Syntax parsing refers to the tree analysis of a given sentence. As mentioned above, we modeled the task in two steps: (1) Annotating the head (parent node) of every word in the sentence; (2) Classifying the syntax relation between each token-head pair. The annotation of the head is done by adding an extra layer on top of the encoder that calculates scores between each pair of tokens similarly to the scoring part in the attention heads. Then after

[9]

calculating the scores, a maximum spanning tree algorithm (McDonald, 2006) is applied to produce the syntax tree. The classification of the syntactic relation was performed by concatenating the vectors of the tokens and pairs predicted by the first step and linearly projecting them to the set of classes. We used *BulTreeBank*'s 8 907 training, 1 115 validation, and 1 116 testing sentences. We train for 5 epochs with linearly decaying learning rate from 5×10^{-5} and batch size of 8*48. The accuracy of the head prediction is referred to as UAS - Unlabeled Attachment Score, while the accuracy of both head prediction and class labeling is referred to as LAS - Labeled Attachment Score. Our best result for the UAS is 0.9571 and the LAS is 0.9330. We report the UAS scores across the pre-trained models in Table 3.

4.2.3. Named Entity Linking

Named Entity Linking is the task of disambiguating names in a text to a knowledge base. In our case, we labeled the names in the text with their Wikipedia article URL. For training, validation, and testing, we used datasets with Wikipedia URL annotation of named entities: *The Bulgarian part of the Balto-Slavic Corpus*, the *BulTreeBank*, and sentences from *The Bulgarian Event Corpus (BEC)*. The resulting set consisted of 53 829 contexts and 7 065 unique named entity urls. We modeled the task as a semantic search. We enclose the entity that needs to be linked with special tokens and treat the whole context as a query. The corresponding Wikipedia article acts as a document. We fine-tuned an encoder for producing embeddings using the Sentence Transformers library (Reimers and Gurevych, 2019). The model was optimized to make the embeddings of the context and the Wikipedia article closer than the random (noise contrasting objective). We split the data into 43 063 training, 5 384 validation, and 5 382 test pairs of entity context and wikipedia article. We train the sentence transformer for 3 epochs with a peak learning rate of 1×10^{-5} and a batch size of 8*16 pairs.
After training, the named entity linking is performed by finding the Wikipedia article with the closest embedding. We perform experiments on the entities with contexts from the test set by disambiguating them to the whole Bulgarian Wikipedia consisting of 151 708 entities. We report Recall at positions: 1, 3, 5, and 10 - 0.83, 0.93, 0.95, and 0.96. The system may benefit from having a cross-encoder model to perform re-ranking, but we leave this for future work.

4.2.4. Word Sense Disambiguation

Word Sense Disambiguation is the task of linking semantically ambiguous words in a text to a thesaurus like WordNet. We modeled the task in two steps: (1) We use a lexicon and a lemmatizer model to find the candidate senses of the word; (2) We use a cross-encoder model to classify the similarity between the context with the word enclosed in special tokens and the definition of the candidate sense from the Bulgarian BTB-WordNet (Simov and Osenova, 2023).
To train the model, we used the same WordNet to create positive and negative data pairs. The BTB-WordNet has example usage for every synset. More specifically, we considered all words that are ambiguous (linked to more than one sense) and created example-definition pairs. The positive pairs were created directly by pairing the sense definition and one of its examples, while the negative ones were taken from the examples of other senses for the same word that is not semantically related. Thus, we created positive and negative pairs. We split the set into 85 254 train, 3 698 validation, and 4 006 test pairs, while making sure that there are no pairs with the same sense in the different partitions, to avoid leakage. We also enhanced the definitions with other words that are linked to the same sense, which improved the accuracy. The models are trained for binary classification in the classes: similar and different. We train for 3 epochs with linearly decaying learning rate from 2×10^{-5} and batch size of 8*32. We report the F1 score over the test set pairs in Table 3.

It resulted in a F1 score of of 82.44% in finding the correct sense of ambiguous words.

4.2.5. Encoder-Decoder conditional generation

As mentioned above, not every task can be modeled as a token classification. For such tasks, more general free form text generation is preferable. Thus, we fine-tuned our T5 models.

Lemmatization is the task of converting a word to its base form (lemma). We modeled it by passing a sentence with words separated by special tokens and training a model to generate the corresponding lemmas in the correct positions. In this way, the whole sentence was lemmatized in a single generation. We fine-tuned subword level T5 model on a train split of our extended version of the *BulTreeBank* corpus and achieved a test accuracy score of 0.9946.

We fine-tuned T5 model for **Spelling correction** of a text with spelling and punctuation errors. We used the same dataset as the text error tagging task described in Subsection 4.2.1. The model achieved a BLEU score of 0.9936 on a test set.

Translation from old spelling is another sequence-to-sequence task. In Bulgaria in the 19th and 20th centuries, there were different spelling systems in comparison to the only one that is used

Model	Tokenization	BLEU (word level)
Lexicon substitution (baseline)		0.8769
T5	Character	**0.9532**
T5	Subword	0.9201
EuroLLM 1.7B	Subword	0.9421

Table 4: BLEU Scores of the tested fine-tuned models for the translation from old spelling task.

in modern Bulgarian. The differences are confined to variations in letter usage. For training, validation and testing data, we used parallel sentences paired from different versions of novels and short stories published in the period of 1910-1944 for which we found also electronic versions published after 1945. We used some heuristics in the alignment, with manual checks on the result for nearly half of the texts. We fine-tuned the models on the dataset for 10 epochs with a peak learning rate of 1×10^{-4}. The best checkpoint according to the validation set is selected. We found that the T5 model that tokenized the text at the character level is better suited than the one that used subword tokenization. The best model achieved a BLEU score of 0.9532 on a test set. We fine-tuned EuroLLM-1.7B (Martins et al., 2025) on the same task and observed that the results are close to the T5 character level results. (Table 4) We hypothesize that the reason is that the tokenizer of EuroLLM, being multilingual, tokenizes old Bulgarian words in smaller subwords, maybe even characters, similarly to the T5 character level.

4.2.6. Event Extraction

Event Extraction is the task of finding and extracting structured information from a given text in the form of events and participants in them. We modeled the task as a sequence-to-sequence generation that takes a sentence as input and produces an output in a JSON format which consists of a list of the events present in the sentence. Each event is represented as an object and has a type, a span - the position in the sentence, and a list of roles - each with a type and a span. We used the *Bulgarian Event Corpus* dataset, splitting it into 13 233 training, 1 471 validation, and 1 634 test sentences. We evaluate the model on the test sentences and report an F1 score of 0.77 in retrieval of event spans and an accuracy of 0.86 in prediction of the type of true positive events.

5. Visualization of the annotations

After accumulating all the above mentioned models for the various tasks, we began developing a user interface to observe the different annotations either in isolation or in combination over documents of interest. For that reason, in this section we present a basic user interface implemented as an HTML editor. This allows the user to observe the documents together with their formatting – different headings, sections, footnotes, bold, italics, etc. Additionally, the HTML mark-up has been extended with information about the available annotation. Here, we consider two scenarios of usage, depending on the user needs and the available corpora. If users want to annotate their own corpora, then it would be useful for them to have the possibility to annotate the corpora with selected types of annotation. However, if the corpora have been shared by many users with different needs, then we imagine that all types of annotation have to be performed. Apart from the NLP tasks discussed above in Sect. 4 we also consider some obvious extensions to the set of NLP tasks to be supported by the set of models. Such a task is for example co-reference resolution.

Having in mind the user requirements, we decided that the best way to encode the annotation information is to represent it on a token level. In this way, the interaction of the HTML mark-up as well as the editing operation with the annotation is maximally reduced. Thus, our goal here is the projection of the corresponding annotation to token level annotation. As is well known, this is possible for many types of annotation. For example, the Universal Dependency (UD) annotations are represented in such a way. In UD we have three main types of annotation: (1) “category” annotation, where the information is represented as a single value, which we interpreted as a category, independently from the fact that these values can be lemmas, tags (representing a bundle of values, positionally), or a list of feature:value pairs; (2) “tree” annotation, where the annotation is represented by arcs between the tokens which form tree structures over the tokens in the text. In this way, UD encodes the dependency structures; (3) “dep-graph” annotation, where the annotation is represented as multiple arcs from a given token to other tokens in the text. UD encodes in a similar way the Enhanced dependency graph, where a list of head-deprel pairs is attached to the tokens. Each head-deprel consists of two elements: the first element is the position of the token in the current sentence, and the second element is the dependency relation. In our work, we would like to extend this annotation to allow the encoding of in-coming and out-leaving arcs, bidirectional arcs, and having different labels.

In order to present more complex graphs than the enhanced dependency graph, we extended the notion of head-deprel to *graph-edge-rel*. Each graph-edge-rel consists of the following elements: *target node* — the position of the token to which the edge

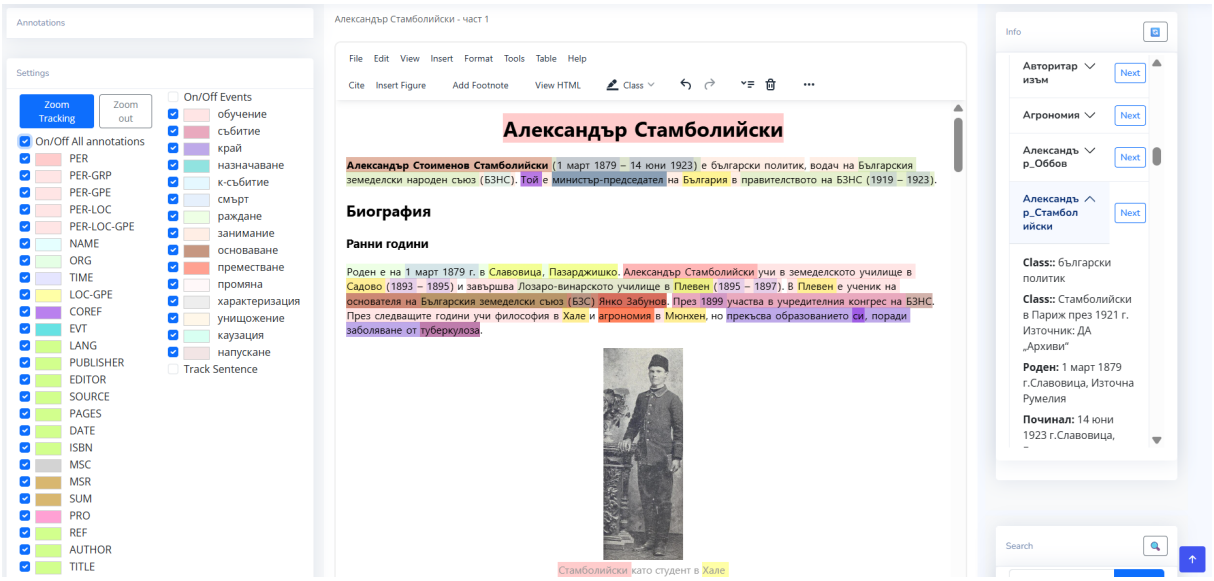

Figure 1: This screenshot depicts the editor main screen containing an annotated document. The annotation scheme has three levels of annotation: Named entity; Named entity linking; Event annotation. Named entity and Named entity linking levels are of type “category”. The Event is represented as a graph between tokens that determines the event in the text. The left part of the screen presents the different categories of Named entities or events. The right part of the screen presents information related to the Named entity links.

is connected; *edge type* — one of the values: *in* (in-coming arc), *out* (out-leaving arc), and *bi* — a bidirectional arc; *label* — the label from a set of labels assigned to the edge. Thus, we want to have an annotation scheme with optional annotations. For example, in the case of the UD CoNLL format we could require having cases in which only XPOS tags are presented.

In order to meet these requirements, for each document, we define an annotation scheme consisting of a list of declarations of the annotation scheme elements. Similarly to CoNLL format, the first element of the annotation scheme is the indexing of the tokens within the text. The token number is global over the whole text, allowing annotations over more than one sentence. Each element declaration in a scheme definition has the form: *type of annotation* — one of the values: “category”, “tree”, “dep-graph”, and “graph”, as defined above; Secondary type of the annotation — one of the values: “single”, “list-of-value”; Set of values — what the values are. The values could be some of the types: “number”, “string”, “tag”

Fig. 1 presents an example of annotated document on three layers of annotation: Named entity layer, Named entity linking layer, and Event layer.

6. Conclusion and Future Work

In this paper, we presented a number of LLMs, fine-tuned to basic NLP tasks for Bulgarian. In our practice, most of them proved to be useful in supporting research in the area of Humanities. To provide the annotations to the researchers, we implemented an approach for encoding the annotations in HTML documents.

We think that all of these tasks are useful for researchers within SS&H. Here, we describe some of the possible scenarios. (1) POS and lemmatization services provide important information for searching in large corpora.[10] Both tasks, mentioned above, are necessary for the preparation of corpora loaded into NoSketch Engine. For most of the users, this information is not necessary. However for linguists, lexicographers, language teachers, and others, the information from the annotation is necessary. (2) WSD service is important for linguists, lexicographers, but also for humanities specialists who are interested in the annotation of their corpora with specific terminological lexicons. (3) NER and NEL services allow for the diverse representation of one or many documents (see the user interface above). (4) UD Parsing and Event services interact for extensions of event extraction within the current event inventory, but also for extensions to new types of events. All of these services are easier to maintain, to extend to new areas, etc. They also significantly contribute to the explainability, and are much cheaper than LLMs with billions

[10] See the massive list of corpora uploaded in NoSketch Engine from CLASSLA: Knowledge centre for South Slavic languages — https://www.clarin.si/ske.

of parameters.

Needless to say, LLMs with billions or even trillions of parameters can also be used. But they are not ready off-the-shelf as many of the users expect. We performed some experiments with ChatGPT services to implement these tasks using the prompting approach. For tasks like POS, Lemmatization, UD Parcing, the results were worse than ours. For the WSD task, both approaches perform similarly. We did not conduct experiments based on fine-tuning of LLMs with billions of parameters. We imagine that such fine-tuned models will be comparable.

In our future work, we plan to extend the set of models to new tasks. We will work on the creation of RAG systems in SS&H in which we consider to use encoder models for vectorization of the document database, and small decoder models for the generation of the answers depending on preselected snippets. Thus, our strategy is to keep the models small but powerful enough for the tasks in hand as much as possible, and to gradually extend the models with respect to architecture and size towards related new tasks.

7. Bibliographical References

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514. ACL.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Viet Dac Lai. 2022. Event extraction: A survey.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Pedro Henrique Martins, Patrick Fernandes, João Alves, Nuno M. Guerreiro, Ricardo Rei, Duarte M. Alves, José Pombal, Amin Farajian, Manuel Faysse, Mateusz Klimaszewski, Pierre Colombo, Barry Haddow, José G.C. de Souza, Alexandra Birch, and André F.T. Martins. 2025. Eurollm: Multilingual language models for europe. *Procedia Comput. Sci.*, 255(C):53–62.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, page 216–220, USA. Association for Computational Linguistics.

Mariana Neves and Jurica Ševa. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163.

Saziye Betül Özates, Arzucan Özgür, Tunga Güngör, and Balkiz Öztürk. 2020. A hybrid approach to dependency parsing: Combining rules and morphology with deep learning. *CoRR*, abs/2002.10116.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019a. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019b. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Siva Reddy, and Aaron Courville. 2021. Explicitly modeling syntax in language models with incremental parsing and a dynamic oracle. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1660–1672, Online. Association for Computational Linguistics.

Étienne Simon, Helene Olsen, Huiling You, Samia Touileb, Lilja Øvrelid, and Erik Velldal. 2024. Generative approaches to event extraction: Survey and outlook. In *Proceedings of the Workshop on the Future of Event Detection (FuturED)*, pages 73–86, Miami, Florida, USA. Association for Computational Linguistics.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021a. Fast WordPiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2089–2103, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yang Song, Xin Cai Ong, Hwee Tou Ng, and Qian Lin. 2021b. Improved Word Sense Disambiguation with Enhanced Sense Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4311–4320. ACL.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference.

Houquan Zhou, Yang Hou, Zhenghua Li, Xuebin Wang, Zhefeng Wang, Xinyu Duan, and Min Zhang. 2023. How well do large language models understand syntax? an evaluation by asking natural language questions.

8. Language Resource References

Marta Bañón, Mălina Chichirău, Miquel Esplà-Gomis, Mikel Forcada, Aarón Galiano-Jiménez, Taja Kuzman, Nikola Ljubešić, Rik van Noord, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Peter Rupnik, Vit Suchomel, Antonio Toral, and Jaume Zaragoza-Bernabeu. 2023. MaCoCu: Massive collection and curation of monolingual and bilingual data: focus on under-resourced languages. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 505–506, Tampere, Finland. European Association for Machine Translation.

Ona de Gibert, Graeme Nail, Nikolay Arefyev, Marta Bañón, Jelmer van der Linde, Shaoxiong Ji, Jaume Zaragoza-Bernabeu, Mikko Aulamo, Gema Ramírez-Sánchez, Andrey Kutuzov, Sampo Pyysalo, Stephan Oepen, and Jörg Tiedemann. 2024. A new massive multilingual dataset for high-performance language technologies. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1116–1128, Torino, Italia. ELRA and ICCL.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4226–4237, Torino, Italia. ELRA and ICCL.

Jakub Piskorski, Bogdan Babych, Zara Kancheva, Olga Kanishcheva, Maria Lebedeva, Michał Marcińczuk, Preslav Nakov, Petya Osenova, Lidia Pivovarova, Senja Pollak, Pavel Přibáň, Ivaylo Radev, Marko Robnik-Sikonja, Vasyl Starko, Josef Steinberger, and Roman Yangarber. 2021. Slav-NER: the 3rd cross-lingual challenge on recognition, normalization, classification, and linking of named entities across Slavic languages. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 122–133, Kiyv, Ukraine. Association for Computational Linguistics.

Kiril Simov and Petya Osenova. 2023. Recent Developments in BTB-WordNet. In *Proceedings of the 12th Global Wordnet Conference*, pages 220–227, University of the Basque Country, Donostia -

San Sebastian, Basque Country. Global Wordnet Association.

Kiril Simov, Petya Osenova, Milena Slavcheva, Sia Kolkovska, Elisaveta Balabanova, Dimitar Doikoff, Krassimira Ivanova, Alexander Simov, and Milen Kouylekov. 2002. Building a linguistically interpreted corpus of Bulgarian: the BulTreeBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Kiril Simov, Nikolay Paev, Petya Osenova, and Stefan Marinov. 2025. Bulgarian event extraction with llms. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI era*, pages 1163–1171, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.