DeepPavlov 1.0: Your Gateway to Advanced NLP Models Backed by Transformers and Transfer Learning

Maksim Savkin[1] **Anastasia Voznyuk**[1] **Fedor Ignatov**[1] **Anna Korzanova**[1]
Dmitry Karpov[1] **Alexander Popov**[1] **Vasily Konovalov**[2,1]
[1]Moscow Institute of Physics and Technology
[2]AIRI, Moscow, Russia
{savkin.mk, vasily.konovalov}@phystech.edu

Abstract

We introduce DeepPavlov 1.0, an open-source framework designed for seamless use of Natural Language Processing (NLP) models, leveraging advanced transfer learning techniques. This framework offers a modular, configuration-based approach, making it suitable for a wide range of NLP applications without requiring in-depth knowledge of machine learning or NLP. Built on PyTorch and supporting Hugging Face `transformers`, DeepPavlov 1.0 provides ready-to-use solutions for various NLP tasks. It is publicly available[1] under the Apache 2.0 license and includes access to an interactive online demo[2].

1 Introduction

Natural Language Processing (NLP) plays a critical role in many AI applications today, facilitating tasks such as automating customer service and processing large volumes of text data. However, the complexity of building, fine-tuning, and deploying state-of-the-art NLP models remains a significant barrier, particularly for non-experts in machine learning. The goal of NLP frameworks is to simplify and streamline the process of solving NLP tasks in software applications. These frameworks offer pre-built components, models, and tools for tasks such as Named Entity Recognition (NER), sentiment analysis, text classification, and more. While the lifespan of NLP frameworks can be limited for various reasons, this does not imply that there is no space for NLP frameworks; instead, it highlights a growing gap in user-friendly, long-term NLP solutions. The continued success of an NLP framework relies not only on offering robust pre-trained models but also on responding to user needs by curating and expanding relevant datasets. Active support and communication with the community provides invaluable feedback and guides development in the right direction.

DeepPavlov (Burtsev et al., 2018b) framework, originally introduced at ACL 2018, was designed to lower the entry barrier for non-experts, making advanced NLP accessible to a wider audience. Originally built on TensorFlow, it has since evolved with the release of DeepPavlov 1.0, which now relies on PyTorch and `transformers` while maintaining full backward compatibility in both philosophy and configuration files. DeepPavlov 1.0 allow to utilize any `AutoConfing` supported models. However, unlike `transformers`, DeepPavlov 1.0 endorses no-code methodology, enabling the training, inference, and deployment exclusively through configuration files. By leveraging `transformers`, DeepPavlov 1.0 supports transfer learning techniques such as Sequential Transfer Learning (STL), Multi-Task Learning (MTL), and Cross-Lingual Transfer Learning (CTL). This enables the development of multilingual models and models that can handle scarce training data effectively.

Throughout its development, we realised that an NLP framework is not merely about pre-trained NLP models; it is equally important to continuously interact with users and address their needs. We concluded that to create a popular NLP framework, we must focus on gathering user-demanded datasets, thus, in Section §5, we discuss the NER dataset that was compiled from openly available NER datasets and additionally augmented with synthetic data generated by Large Language Models (LLMs). This dataset included custom location-related entities such as `region`, `city`, `street_name`, `building_number`, and `apartment`. The selection of entities was guided by user feedback, which was gathered through surveys conducted on our forum.

Our contribution can be summarized as follows:

- We have developed an open-source NLP

[1]https://github.com/deeppavlov/DeepPavlov
[2]https://demo.deeppavlov.ai

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing System Demonstrations, pages 465–474
November 12-16, 2024 ©2024 Association for Computational Linguistics

framework, DeepPavlov 1.0, which is grounded in PyTorch and utilizes `transformers`. It supports a variety of fine-tuned models that address a wide range of NLP tasks.

- We have developed an accessible and user-friendly infrastructure, optimized for both novice users and those seeking production-ready applications.

- We have established a community around DeepPavlov, featuring a forum, interactive demos, and continuous user support.

2 Related Work

Numerous frameworks are available to facilitate the development of NLP models. Before discussing related work and comparable frameworks, it is important to first clarify what is similar but not directly relevant.

Although PyTorch and TensorFlow allow users to build NLP models, they require expertise in ML/NLP for training and deployment, placing them in a different category. While Keras, PyTorch Lightning and transformers require less ML design knowledge from users, they still lack production-ready NLP models and tools like Docker and API interfaces, making them distinct from our focus.

One could argue that in the realm of LLMs there might be less need for traditional NLP frameworks. Models like ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) demonstrate remarkable performance across a broad spectrum of NLP tasks. However, they have significant downsides and limitations. For instance, ChatGPT often struggles with arithmetic, spatial, temporal, physical, and logical reasoning (Borji, 2023). Additionally, deploying LLMs requires costly specialised hardware that many organizations cannot readily acquire. Privacy concerns also arise, as sharing sensitive data with LLM providers may violate user agreements or federal laws. In addition, encoder-based models fine-tuned with adequate training data typically surpass LLMs in a majority of NLP tasks, including NER (Hu et al., 2024), sentiment analysis (Wang et al., 2023), and QA (Li et al., 2023). These limitations restrict the use of LLMs in real-world production environments. Hence, frameworks like LangChain[3] and others will not be included in our comparison for this work.

[3] `https://langchain.com`

A comparison of DeepPavlov 1.0 with other NLP frameworks can be found in Table 1.

spaCy[4] (Honnibal and Montani, 2017) has consistently been an industry-standard NLP framework since its initial release in February 2015. The framework has a huge ecosystem: the recent version supports 72+ languages and 80 trained pipelines for 24 languages. `spaCy`'s transformer architectures ensure compatibility with PyTorch and the HuggingFace's `transformers` library, granting users access to thousands of pre-trained models for integration into their pipelines. `spaCy` is an irreplaceable tool for extracting linguistic characteristics, such as POS tagging, morphology, lemmatization and more. Like DeepPavlov, it supports MTL using pre-trained transformers.

Flair[5] (Akbik et al., 2019) initially provided a straightforward and unified interface for various conceptually distinct types of word and document embeddings. As a result, it has emerged as a highly popular NLP framework for tasks such as NER, POS tagging, sentiment analysis, entity linking, and more. It accomplished this by supporting highly performing models, some of which were backed by less accurate yet lightweight RNN-based architectures.

Furthermore, Flair has become a go-to testbed for NLP research, with projects such as FLERT (Schweter and Akbik, 2020) and TARS (Halder et al., 2020) relying on it for evaluation.

Stanza[6] (Qi et al., 2020) features a native Python interface to the popular Java-based Stanford CoreNLP (Manning et al., 2014) software, thus expanding its capabilities to include tasks such as coreference resolution and relation extraction. Similarly to traditional NLP frameworks, Stanza offers models for tokenization, lemmatization, POS tagging, dependency parsing, and NER. Furthermore, Stanza has been applied to biomedical and clinical text analysis, particularly for syntactic processing and NER tasks in these domains (Zhang et al., 2021).

Below, we list related frameworks that are no longer supported:

jiant[7] (Phang et al., 2020) is a multi-task and transfer learning toolkit for NLP research that was initiated at NYU CILVR. `jiant` is very similar

[4] `https://github.com/explosion/spaCy`
[5] `https://github.com/flairNLP/flair`
[6] `https://stanfordnlp.github.io/stanza`
[7] `https://github.com/nyu-mll/jiant`

Framework	Online Demo	Docker	API	CLI	MTL	Maintenance Status	GitHub Stars, $\cdot 10^3$	Licence
DeepPavlov 1.0	✓	✓	✓	✓	✓	Active (2024)	6	Apache-2.0
spaCy	✓	✓	✓	✓	✓	Active (2024)	29	MIT
Stanza	✓	✗	✗	✗	✗	Active (2024)	7	Apache-2.0
Flair	✓	✗	✗	✗	✗	Active (2023)	13	MIT
AllenNLP	✓	✓	✓	✓	✓	Inactive (2022)	11	Apache-2.0
jiant	✗	✗	✗	✗	✗	Inactive (2021)	1	MIT

Table 1: Comparison of NLP frameworks. CLI – Command Line Interface, MTL – Multi-Task Learning. The "Maintenance Status" indicates whether the framework is currently active or inactive, with the year of the last release noted in brackets. For more details, please refer to Section 2.

to DeepPavlov in a sense that it supports multiple transfer learning techniques, such as sequential training and multi-task training, and also integrates with `datasets` and `transformers` to manage models and data. Additionally, `jiant` supports GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019) and XTREME (Hu et al., 2020) benchmarks. Unfortunately, since October 17, 2021, the `jiant` project is no longer actively maintained.

AllenNLP[8] (Gardner et al., 2018) is an open-source NLP library, created by AI2. Initially, it gained widespread adoption due to its inclusion of pretrained ELMo (Peters et al., 2018a) representations. As a result, it emerged as a popular NLP framework for tasks such as text understanding, information extraction, sentiment analysis, and reading comprehension. However, it is worth noting that the AllenNLP repository was archived and ownership transferred on December 16, 2022.

3 Design and Implementation

DeepPavlov 1.0 adheres to the core model organization schema inherited from its predecessor, DeepPavlov (Burtsev et al., 2018a). To make it easier for newcomers, NLP models in DeepPavlov are defined in separate configuration files, which include the parameters needed for training, inference, and deployment: `dataset_reader`, `dataset_iterator`, `chainer`, `train`, and `metadata`.

Sections `dataset_reader` and `dataset_iterator` are responsible for accessing the data and splitting it into training, validation, and test sets. `dataset_reader` supports `datasets` from HuggingFace.

[8]https://github.com/allenai/allennlp

The `chainer` is a core concept of DeepPavlov: it builds a pipeline from heterogeneous components (Rule-Based/ML/DL) and makes it possible to train or infer the entire pipeline as a unified unit. In addition, `chainer` specifies component inputs (`in`, `in_y`) and outputs (`out`) as arrays of names.

A pipeline element can be either a function or an object of a class that implements `__call__` method. Any configuration file can be used within another configuration file as an element of the `chainer`, and any field of the nested configuration file can be overwritten.

The `train` section defines training hyperparameters, such as trainer class, evaluation metrics, batch size, early stopping criteria, and many others.

The `metadata` section contains variables used in other sections of the configuration file, for example, transformer encoder `AutoConfig` alias, paths to pretrained model and corresponding dataset.

DeepPavlov allows users to easily customize the model configuration file. They can modify the hyperparameters, change the data preprocessing, or switch the classification model in the `chainer` while keeping the input and output format intact. For example, training the model on your own dataset is straightforward: just set the `data_path` in the `dataset_reader` to desired dataset location, and, if needed, adjust the training hyperparameters in the `train` section.

A generalized schema of a DeepPavlov's configuration file is depicted in Figure 1.

4 Usage

DeepPavlov 1.0 is developed in Python and utilizes PyTorch as the base machine learning framework that facilitates multi-GPU training by leverag-

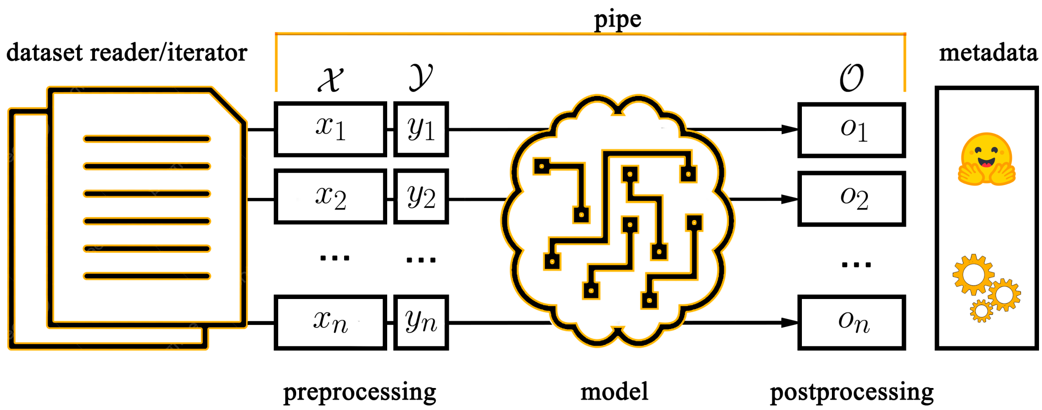

Figure 1: Overview of DeepPavlov's training configuration file. Dataset reader is responsible for reading files, dataset iterator splits data into batches, and the pipe consists of various processing steps. Also, one can customise the training processing with the metadata section of the file.

ing PyTorch's `DataParallel`. DeepPavlov 1.0 incorporates HuggingFace `transformers`, enabling the utilization of all `AutoModel` transformer-based models from the HuggingFace Hub. The framework offers versatile interaction with models through a command-line interface (CLI), representational state transfer (REST), application programming interface (API), or Python. DeepPavlov 1.0 can be installed by running `pip install deeppavlov`. The list of supported CLI commands is as follows:

install To install model-specific requirements, run `python -m deeppavlov install <config_name>`, where `<config_name>` is the name of the configuration file.

interact To get predictions from a model interactively through CLI, run `python -m deeppavlov interact <config_name> [-d] [-i]`, where `-d` downloads files from the `metadata` of the configuration file (optional), and `-i` installs model requirements (optional).

train To start the model training process, run the following command: `python -m deeppavlov train <config_name> [-d] [-i]`. The dataset will be downloaded regardless of whether there was the `-d` flag or not. To train on custom data, users need to modify the `dataset_reader` path in the model configuration file. The data format is specified on the corresponding model documentation page. To change the architecture of the backbone transformer, users should modify the corresponding variable in the `variables` section.

riseapi To run REST-like API server with the selected model, which might be useful for production usage, run `python -m deeppavlov riseapi <config_name> [-d] [-i]`.

Appendix A contains a code snippet that demonstrates how to interact with DeepPavlov models.

5 Reference Models

DeepPavlov 1.0 offers a range of fine-tuned NLP models for tasks such as text classification, token classification, and question answering. Although many of the frameworks listed in Table 1 accommodate more specialised models such as syntax parsing, lemmatization, and dependency parsing, our focus lies on more practical models like NER, sentiment analysis, and QA. The selection of these models was driven by user demand. We surveyed users and reviewed model download stats to find the most important areas for further development. It became clear that over half of the downloaded models are for named entity recognition. Reading comprehension and text classification also rank prominently among the frequently accessed tasks. A complete list of supported models is available in our documentation[9].

Text Classification component of the DeepPavlov 1.0 framework enables to perform sentiment analysis, toxicity identification, topic classification. Our unique topic classification model is fine-tuned on the "DeepPavlov Topics" dataset for conversational domain (Sagyndyk et al., 2023), which consists of 33 topics with 4.2M instances in total. The dataset was automatically collected and filtered from websites and open datasets. The

[9] `http://docs.deeppavlov.ai`

Framework	Model	OntoNotes 5.0	CoNLL '03
DeepPavlov 1.0	DeBERTa-v3-base	90.3	93.1
Flair (large)	XLM-RoBERTa-large	**90.9**	**94.1**
spaCy	RoBERTa-base	89.8	91.6
Stanza	LSTM	88.8	92.1
Flair (base)	Flair embeddings (Akbik et al., 2018)	89.7	93.1
DeepPavlov	Bi-LSTM-CRF (The et al., 2017)	86.7	89.9
AllenNLP	Bi-LSTM-CRF+ELMo (Peters et al., 2018b)	–	92.2

Table 2: Performance comparison of NER models on OntoNotes and CoNLL datasets. Entity micro-averaged F1-score is reported. Here, Flair (large) marginally outperforms DeepPavlov 1.0, which is based on DeBERTa-v3-base. However, it's worth noting that XLM-RoBERTa-large has over 500M parameters, whereas DeBERTa-v3-base has only 86M backbone parameters. This makes DeepPavlov's model more efficient.

DeepPavlov topic classifier[10] is based on *distilbert-base-uncased* encoder model, which saves computational resources and speeds up the inference time (Kolesnikova et al., 2022).

Token Classification is traditionally represented by NER and PoS models. Based on our surveys, NER is the most popular model within DeepPavlov. Most NLP frameworks include NER models fine-tuned on openly available datasets, such as CoNLL-2003[11] (Tjong Kim Sang and De Meulder, 2003) and OntoNotes[12] (Pradhan et al., 2013).

Table 2 presents a comparison of NER models from various frameworks that are pre-trained on open NER datasets. However, most open datasets often do not encompass the full range of entities required by users.

Based on user surveys, we consistently expand the range of NER entities by integrating data sets or generating synthetic data. Our primary DeepPavlov 1.0 NER model[13] includes 32 entity types. The entity taxonomy is illustrated in Figure 3. The final NER model was trained on a combination of the OntoNotes 5.0 and Massive (FitzGerald et al., 2023) datasets. Furthermore, we expanded these datasets with synthetic texts containing fine-grained location-related entity types requested by our users, specifically `street_name`, `state`, `region`, `city`, `building_number` and `apartment`.

To generate location-related entities, we employed OpenChat-3.5-0106 (7B) (Wang et al., 2024). Using multiple prompt variations, we created a diverse collection of texts and manually inspected them for any inconsistencies in the automatic markup. Next, we merged the open datasets with synthetic texts by training two encoder-based models on each part individually and labelling the remaining parts. Following the merge, we obtained a unified dataset incorporating all labels from both initial datasets. The DeepPavlov 1.0 NER demo features 32 entities, which exceeds the number of entities in any other freely available NER dataset, to our knowledge, see Table 3.

Dataset	# Types	Ln(s)
DeepPavlov$_{\text{NER}}$	32	en, ru
kazNERD	25	kk
OntoNotes 5.0	18	en, zh, ar
ARMTDP	18	hy
CoNLL 2003	4	en, de

Table 3: The number of entity types in some of the more popular open-source NER datasets. DeepPavlov$_{\text{NER}}$ – represent custom NER models provided by DeepPavlov 1.0.

Along with a standard sentiment classification model, we developed an Aspect-Based Sentiment Analysis (ABSA) (Liu, 2012) model in response to our users' requests. ABSA is a refined form of sentiment analysis that identifies aspects and their corresponding opinions within a given text. It has gained significant popularity in marketing because it offers more nuanced and targeted insights. Specifically, we perform aspect-sentiment pair extraction (ASPE) based on the input sequence S, where we label each token S_i with sentiment polar-

[10] `topics_distilbert_base_uncased`
[11] `ner_conll2003_bert`
[12] `ner_ontonotes_bert`
[13] `ner_bert_base`

ities if applicable. In the example, *'I really like the interior but am disappointed with the dynamics of this car'*, *'interior'* would be labelled as `positive`, and *'dynamics'* as `negative`.

The ASPE formulation resembles a NER task with four classes: `not an aspect`, `negative`, `neutral`, and `positive`. As the base for our training set we utilized the SemEval-2014 Task 4 dataset (Pontiki et al., 2014), encompassing multi-domain data pertinent to ABSA. Our proposed model attained micro-F1 scores of 68.8% and 80.5% for the laptop domain and restaurant domain accordingly. Furthermore, as illustrated by the comparative analysis in Wang et al. (2023), our methodology consistently surpasses ChatGPT's performance across all evaluated scenarios.

To our knowledge, none of the listed NLP frameworks in Table 1 support an ABSA model.

Reading Comprehension is underrepresented in NLP frameworks, yet it is crucial for developing ODQA (Weng, 2020) systems, especially when there is a lack of resources for deploying a LLM with Retrieval-Augmented Generation (RAG). DeepPavlov 1.0 improved DeepPavlov's performance on the SQuAD 1.1 (Rajpurkar et al., 2016) validation set, achieving an exact match score of 81.49% (an increase from 80.88%) and an F1-score of 88.86% (up from 88.49%).

Multi-task learning in DeepPavlov 1.0 supports the following tasks types: multiple choice classification, text classification, text regression, text binary scoring, and token classification. The MTL implementation is transformer-based and encoder-agnostic: any `AutoModel`[14] can be used in its pipeline. In the multi-task model, one backbone transformer that is the same for all tasks extracts features from the input text. Then, for every task, these features are processed by the task-specific linear layer to obtain the predictions (Karpov and Konovalov, 2023). We also show MTL results on the GLUE benchmark in Appendix C.

6 Applications

Dream dialog system (Zharikova et al., 2023) leverages DeepPavlov MTL model fine-tuned on five classification tasks: emotion, sentiment, toxicity, intent, and topic. Furthermore, the DeepPavlov's 1.0 NER model was enhanced to specifically handle issues related to the truecasing of Automatic Speech Recognition (ASR) output, thereby facilitating its integration into a virtual assistant pipeline (Chizhikova et al., 2023).

In addition, we have been using DeepPavlov 1.0 for fast prototyping while participating in SemEval competitions, particularly, in Multilingual Shared Task on Hallucinations and Related Observable Overgeneration Mistakes (SHROOM) (Maksimov et al., 2024) and in Machine-Generated Text Detection (Voznyuk and Konovalov, 2024).

7 Conclusion and Future Work

We introduced DeepPavlov 1.0, an open-source NLP framework designed to streamline NLP model development and deployment. Built on PyTorch and transformers, DeepPavlov 1.0 offers a diverse range of pre-trained NLP models accessible through API, CLI, and Python bindings. DeepPavlov 1.0 effectively addresses the challenge of limited training data by using transfer learning.

This framework allows developers to focus on high-level implementation, reducing the need for extensive technical intricacies. To further assist users, DeepPavlov 1.0 boasts an interactive online demo[15] and a dedicated forum[16] for support and collaboration.

During the development of DeepPavlov 1.0, we placed a strong emphasis on user needs. In our ongoing efforts, our aim is to enhance the framework by further expanding the NER model with custom entities. Furthermore, we plan to construct a multilingual ASPE dataset, broadening the scope and usability of DeepPavlov 1.0.

Limitations

In this section, we underscore a significant limitation of DeepPavlov 1.0. During development, we mainly concentrated on improving a small number of top models favored by our users, which were selected through continuous user studies. Consequently, we do not offer models for tasks such as Part-of-Speech (POS) tagging or Dependency parsing, as they did not make the cut according to user preferences. Our philosophy is to excel in supporting a limited range of high-demand models rather than attempting to cover the entire spectrum of NLP models without adequate backing.

DeepPavlov's 1.0 evolution has been guided by the assumption that its primary use case involves

[14] https://hf.co/docs/transformers/model_doc/auto

[15] https://demo.deeppavlov.ai

[16] https://forum.deeppavlov.ai

the application of pre-trained models. As a result, considerable effort was dedicated to enhancing the quality of these pre-trained models. Although DeepPavlov 1.0 does accommodate model fine-tuning, this process demands a level of expertise in NLP and programming knowledge from the user.

For the majority of our models, we do not attain state-of-the-art (SOTA) performance on popular benchmarks. This is due to our commitment to striking a balance between model performance, inference speed, and resource requirements. Typically, SOTA models are resource-intensive, making them impractical for many of our users. Our aim is to provide models that strike an optimal balance between these factors, prioritizing usability and efficiency alongside performance.

Ethical Considerations

We meticulously supervised the generation of the training dataset for our NER model, and we can confirm that no inappropriate or offensive content was found within it.

Acknowledgments

This work was supported by a grant for research centers, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730324P540002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: an easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 54–59. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics.

Ali Borji. 2023. A categorical archive of chatgpt failures. *CoRR*, abs/2302.03494.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, and Vasily Konovalov. 2018a. Deeppavlov: An open source library for conversational ai. In *NIPS*.

Mikhail Burtsev, Alexander V. Seliverstov, Rafael Airapetyan, Mikhail Y. Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018b. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 122–127. Association for Computational Linguistics.

Anastasia Chizhikova, Vasily Konovalov, and Mikhail Burtsev. 2023. Multilingual case-insensitive named entity recognition. In *Advances in Neural Computation, Machine Learning, and Cognitive Research VI*, pages 448–454, Cham. Springer International Publishing.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gökhan Tür, and Prem Natarajan. 2023. MASSIVE: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4277–4302. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3202–3213. International Committee on Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. 7(1):411–420.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080.

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, and Hua Xu. 2024. Improving large language models for clinical named entity recognition via prompt engineering. *J. Am. Medical Informatics Assoc.*, 31(9):1812–1820.

Dmitry Karpov and Vasily Konovalov. 2023. Knowledge transfer in the multi-task encoder-agnostic transformer-based models. *Computational Linguistics and Intellectual Technologies*.

Alina Kolesnikova, Yuri Kuratov, Vasily Konovalov, and Mikhail Mikhail. 2022. Knowledge distillation of russian language models with reduction of vocabulary. In *Computational Linguistics and Intellectual Technologies*. RSUH.

Xianzhi Li, Samuel Chan, Xiaodan Zhu, Yulong Pei, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. 2023. Are chatgpt and GPT-4 general-purpose solvers for financial text analytics? A study on several typical tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023*, pages 408–422. Association for Computational Linguistics.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):58–90.

Ivan Maksimov, Vasily Konovalov, and Andrei Glinskii. 2024. DeepPavlov at SemEval-2024 task 6: Detection of hallucinations and overgeneration mistakes with an ensemble of transformer-based models. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 274–278, Mexico City, Mexico. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `http://openai.com/blog/chatgpt`.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Jason Phang, Phil Yeres, Jesse Swanson, Haokun Liu, Ian F. Tenney, Phu Mon Htut, Clara Vania, Alex Wang, and Samuel R. Bowman. 2020. `jiant` 2.0: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 27–35. The Association for Computer Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 101–108. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Beksultan Sagyndyk, Dilyara Baymurzina, and Mikhail Burtsev. 2023. Deeppavlov topics: Topic classification dataset for conversational domain in english. In *Advances in Neural Computation, Machine Learning, and Cognitive Research VI*, pages 371–380, Cham. Springer International Publishing.

Stefan Schweter and Alan Akbik. 2020. FLERT: document-level features for named entity recognition. *CoRR*, abs/2011.06993.

Anh Le The, Mikhail Y. Arkhipov, and Mikhail S. Burtsev. 2017. Application of a hybrid bi-lstm-crf model to the task of russian named entity recognition. *CoRR*, abs/1709.09686.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Anastasia Voznyuk and Vasily Konovalov. 2024. DeepPavlov at SemEval-2024 task 8: Leveraging transfer learning for detecting boundaries of machine-generated texts. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1821–1829, Mexico City, Mexico. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355. Association for Computational Linguistics.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2024. Openchat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023. Is chatgpt a good sentiment analyzer? A preliminary study. *CoRR*, abs/2304.04339.

Lilian Weng. 2020. How to build an open-domain question answering system? *lilianweng.github.io*.

Yuhao Zhang, Yuhui Zhang, Peng Qi, Christopher D. Manning, and Curtis P. Langlotz. 2021. Biomedical and clinical english model packages for the stanza python NLP library. *J. Am. Medical Informatics Assoc.*, 28(9):1892–1899.

Diliara Zharikova, Daniel Kornev, Fedor Ignatov, Maxim Talimanchuk, Dmitry Evseev, Ksenya Petukhova, Veronika Smilga, Dmitry Karpov, Yana Shishkina, Dmitry Kosenko, and Mikhail Burtsev. 2023. Deeppavlov dream: Platform for building generative AI assistants. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2023, Toronto, Canada, July 10-12, 2023*, pages 599–607. Association for Computational Linguistics.

A Python Code to Interact with DeepPavlov's Model

```
!pip install deeppavlov
!python -m deeppavlov install ner_bert_base
from deeppavlov import build_model
ner_model = build_model('ner_bert_base',
  download=True, install=True)
ner_model(['Bob Ross lived in Florida',
  'Elon Musk founded Tesla'])
# Train model on the other huggingface AutoModel backbones
from deeppavlov import train_model
from deeppavlov.core.commands.utils import parse_config
model_config = parse_config('ner_bert_base')
model_config['metadata']['variables']['BASE_MODEL']=
'distilbert/distilbert-base-cased'
ner_model = train_model(model_config)
# To get predictions in an interactive mode through CLI
!python -m deeppavlov interact ner_bert_base -d
# Make model available for inference as a REST web service
!python -m deeppavlov riseapi ner_bert_base -d
```

Figure 2: Python to interact with DeepPavlov's NER.

B DeepPavlov 1.0 NER Dataset

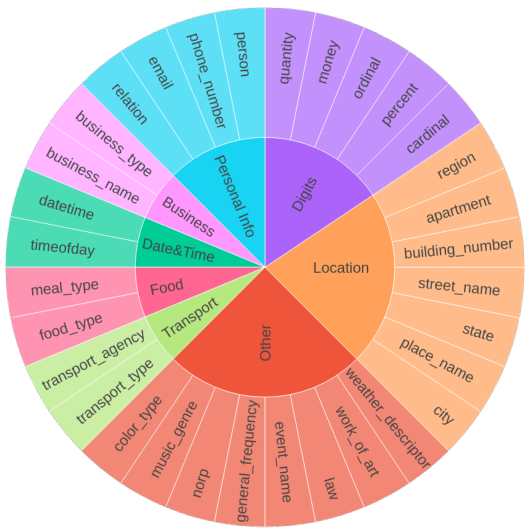

Figure 3: Distribution of entities in DeepPavlov 1.0 NER dataset

C DeepPavlov Multi-Task Learning Results

Model	Mode	Average metric	CoLA M.Corr	SST-2 Acc	MRPC F1/Acc	STS-B P/S Corr	QQP F1/Acc	MNLI Acc(m/mm)	QNLI Acc	RTE M.Corr	AX
Human baseline	-	87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	-
distilbert-base-uncased	S	**75.4**	**47.6**	**92.4**	**86.8**/82.7	81.7/80.7	69.0/87.5	**82.4/81.3**	**88.8**	58.2	33.0
	M	74.7	28.1	91.8	86.7/**87.2**	**83.8/83.0**	**70.2/89.1**	81.0/80.6	88.5	**70.9**	**33.8**
bert-base-uncased	S	77.6	**53.6**	92.7	87.7/83.6	**84.4**/83.1	**70.5/88.9**	**84.4/83.2**	90.3	63.4	**36.3**
	M	**77.8**	43.6	**93.2**	**88.6/84.2**	84.3/**84.0**	70.1/87.9	83.0/82.6	**90.6**	**75.4**	35.4
bert-large-uncased	S	**79.1**	**55.6**	**93.5**	**88.8/84.1**	**86.0/84.9**	70.7/**89.0**	**85.7/85.6**	**92.3**	68.2	38.0
	M	78.8	49.4	93.4	87.4/83.1	84.1/83.7	**71.0**/88.6	85.1/83.9	90.7	**77.4**	**39.3**

Table 4: Metrics of the DeepPavlov's MTL model for the GLUE benchmark. M.Corr stands for Matthew's correlation, P/S corr stands for Pearson/Spearman correlation, Acc stands for accuracy, m/mm means "matched/mismatched", mode S stands for singletask, and mode M stands for multi-task. Results show that multi-task models either approach the metrics of analogous singletask models or even exceed them.