DeepPavlov: Open-Source Library for Dialogue Systems

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail 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, Marat Zaynutdinov

Moscow Institute of Physics and Technology /
9 Institutskiy per., Dolgoprudny, 141701, Russian Federation
burtcev.ms@mipt.ru

Abstract

Adoption of messaging communication and voice assistants has grown rapidly in the last years. This creates a demand for tools that speed up prototyping of feature-rich dialogue systems. An open-source library DeepPavlov is tailored for development of conversational agents. The library prioritises efficiency, modularity, and extensibility with the goal to make it easier to develop dialogue systems from scratch and with limited data available. It supports modular as well as end-to-end approaches to implementation of conversational agents. Conversational agent consists of skills and every skill can be decomposed into components. Components are usually models which solve typical NLP tasks such as intent classification, named entity recognition or pre-trained word vectors. Sequence-to-sequence chit-chat skill, question answering skill or task-oriented skill can be assembled from components provided in the library.

1 Introduction

Dialogue is the most natural way of interaction between humans. As many other human skills are already being mastered by machines, meaningful dialogue is still a grand challenge for artificial intelligence research. Conversational intelligence has multiple real-world applications. Dialogue systems can significantly ease mundane tasks in technical support, online shopping and consulting services.

However, at the moment the research and development in dialogue systems and chatbots are hampered by the scarcity of open-source baselines and impossibility to effectively reuse existing code in new solutions. Therefore, in order to improve upon state-of-the-art dialogue models one needs to implement such a system from scratch. This slows down the progress in the field. In order to overcome this limitation we create **DeepPavlov**[1] — an open-source library for fast development of dialogue systems. DeepPavlov is designed for:

- development of production-ready chatbots and complex conversational systems;
- research in dialogue systems and NLP in general.

Our goal is to enable AI application developers and researchers with:

- a set of pre-trained NLP models, pre-defined dialogue system components (ML/DL/Rule-based) and pipeline templates;
- a framework for implementing and testing their own dialogue models;
- tools for integration of applications with adjacent infrastructure (messengers, helpdesk software etc.);
- a benchmarking environment for conversational models and uniform access to relevant datasets.

The library has a wide range of state-of-the-art solutions for NLP tasks which are used in dialogue systems. These NLP functions address low-level tasks such as tokenisation and spell-checking as well as a more complex, e.g. recognition of user intents and entities. They are implemented as modules with unified structures and are easily combined into a pipeline. A user of library also has a set of pre-trained models for easy start. A

[1] https://github.com/deepmipt/DeepPavlov

model that suits user's task best can be adapted and fine-tuned to achieve required performance. Unlike many other frameworks, DeepPavlov allows combining trainable components with rule-based components and neural networks with non-neural ML methods. In addition to that, it allows end-to-end training for a pipeline of neural models.

The paper is organised as follows. In section 2 we review the existing NLP libraries and explain how they differ from our work. Section 3 describes architecture of DeepPavlov, and in section 4 we talk about features which are available for user of the library and ways of extending it. Section 5 presents some components of the library and benchmarks. Finally, in section 6 we conclude and outline directions for future work.

2 Related work

One of the closest analogues of DeepPavlov is **Rasa Stack** [2] tool. In terms of purpose it is similar to our library. It provides building blocks for creating dialogue agents: natural language understanding, dialogue state tracking and policy. Rasa's capabilities are mainly focused on task oriented dialogue, so unlike our library, it is not readily applicable for constructing agents with multiple skills including chit-chat. It is also important that Rasa Stack exports ML components from other libraries and DeepPavlov includes its' own models. That makes easier for developers to fit trainable parts of the system to the task at hand or add custom ML models. In addition to that, DeepPavlov is more general, and allows defining any NLP pipeline, not only the one related to task oriented dialogue.

Another framework for dialogue agents is **ParlAI** (Miller et al., 2017). ParlAI is in essence a collection of dialogue datasets and models. It defines standard interfaces for accessing the data, provides instruments for training models with any registered dataset and easy integration with Amazon Mechanical Turk. ParlAI does not have any restrictions on models which are implemented there. The only requirement is to support the standard interface. This enables efficient sharing, training and testing dialogue models. Alternatively, in DeepPavlov all agents, skills and models must have a standard structure to ensure reusability.

OpenDial[3] is a toolkit for developing spoken dialogue systems. It was designed to perform dialogue management tasks, but then extended for building full-fledged dialogue systems, integrating speech recognition, language understanding, generation, speech synthesis, multimodal processing and situation awareness. OpenDial includes a number of advanced components but lacks recent deep learning models. Unfortunately ecosystem of deep learning models in Python is not easily accessible from OpenDial because it is Java-based.

	DeepPavlov	Rasa	ParlAI	spaCy	AllenNLP	Stanford NLP
Dialogue systems (DS) features						
Modular architecture of DS	X	X				
Framework for training and testing DS	X		X			
Collection of datasets and DSs			X			
Interactive data labeling and training		X		X		
Integration with messaging platforms	X		X			
Dialogue manager	X	X				
Slot filling	X	X				
NLP features						
Text pre-processing	X	X	X	X	X	X
Word embedding	X	X	X	X	X	X
Intent recognition	X	X		X		X
Entity recognition	X	X		X		X
POS tagging	X			X		X
Dependency parsing				X	X	X
Semantic role labelling					X	X
Sentence embedding			X		X	

Table 1: Comparison of DeepPavlov with other libraries and frameworks.

AllenNLP (Gardner et al., 2017) is another example of a powerful NLP framework. It contains numerous solutions for NLP tasks, but does not include any dialogue models yet. Tailored for NLP research deep learning components of AllenNLP implemented in PyTorch (Paszke et al., 2017) library, which is more convenient for research, then for industrial applications. On the other hand, DeepPavlov by default uses TensorFlow[4] production grade machine learning framework. Another limitation of AllenNLP is the fact that it has only neural models, whereas in DeepPavlov it is possible to combine in a single pipeline heterogeneous components, such as rule-based modules, non-neural ML models and neural networks.

General NLP frameworks can be also used for development of dialogue systems, as they provide low-level operations such as tokenisation, lemmatisation, part-of-speech tagging and syntactic pars-

[2] `http://rasa.com/products/rasa-stack/`
[3] `http://www.opendial-toolkit.net/`

[4] `https://www.tensorflow.org/`

ing. The most notable examples of such frameworks are **Stanford CoreNLP** (Manning et al., 2014) and **spaCy**[5]. Both frameworks provide a set of pre-trained NLP models and functionality for training but have no specific tools and components related to the development of dialogue systems. Stanford tools are in Java, which complicates their integration with a trove of deep learning models in Python.

Table 1 gives comparison of DeepPavlov with other related frameworks.

3 Architecture

The high-level architecture of the library is shown in figure 1. It has several core concepts.

The smallest building block of the library is `Model`. `Model` stands for any kind of function in an NLP pipeline. It can be implemented as a neural network, a non-neural ML model or a rule-based system. Besides that, `Model` can have nested structure, i.e. a `Model` can include other `Model`(s). The library currently has models for intent classification, entity recognition, dialogue state tracking, spell-checking and ranking of texts by similarity.

Models can be joined into a `Skill`. `Skill` solves a larger NLP task compared to `Model`. However, in terms of implementation `Skills` are not different from `Models`. The only restriction for `Skills` is that their input and output should both be strings. Therefore, `Skills` are usually associated with dialogue tasks. There are currently three `Skills` implemented in the library, namely, modular and sequence-to-sequence goal-oriented skills as well as question answering module.

Finally, the core concept of the library is an `Agent`. `Agent` is supposed to be a multi-purpose dialogue system that comprises several `Skills` and can switch between them. It can be a dialogue system that contains a goal-oriented and chatbot skills and chooses which one to use for generating the answer depending on user input.

The choice of `Skill` relevant to the current dialogue state is managed by a `Skills Manager`. This is similar to architecture of Microsoft Cortana (Sarikaya et al., 2016) where *Experience providers* correspond to `Skills`, and selection between them is conducted by a separate module based on context and providers' responses. Systems with multiple skills and their dynamic selection are state of the art in development of dialogue agents, but there is currently no available implementations of such technique.

`Models` are joined in a `Skill` via `Chainer`. `Chainer` takes configuration file in JSON format and sets parameters of `Models` and the order of their execution. Joining heterogeneous models is a striking feature of DeepPavlov library which distinguishes it from other frameworks. Unlike AllenNLP or Tensor2Tensor where all adjacent models need to be neural, in DeepPavlov the pipeline can include neural networks, other ML models, and rule-based models.

4 Usage

The DeepPavlov library is implemented in Python 3.6 and uses `Keras` and `TensorFlow` frameworks. It is open-source and available on GitHub under Apache 2.0 license.

A typical use scenario is the following. A developer takes a pre-build agent, for example, a modular task-oriented bot, and adapts it to the target task. Alternatively, an agent can be built from scratch. In this case skills or models are selected from available `Skills` and `Models`, or created by developer. The models which are included into the agent are trained according to a pipeline defined in a JSON file. DeepPavlov has a collection of pre-trained models, so training is not needed in many cases.

4.1 Training

DeepPavlov supports **end-to-end training**. `Models` implemented on top of `TensorFlow` can be stacked and trained jointly. This feature is sought after in many NLP tasks, in particular in goal-oriented dialogue systems. Usually task-oriented modular systems consist of independently trained building blocks, such as, natural language understanding module, user intent classification module, dialogue policy manager, etc. (Chen et al., 2017). There exist efforts of training such systems in the end-to-end mode (Li et al., 2018). However, such works are difficult to replicate and build upon because of lack of open implementations of end-to-end training. To the best of our knowledge, DeepPavlov is the only NLP framework which allows easy and configurable end-to-end training of dialogue agents created from interchangeable functional neural network blocks.

[5] `https://spacy.io/`

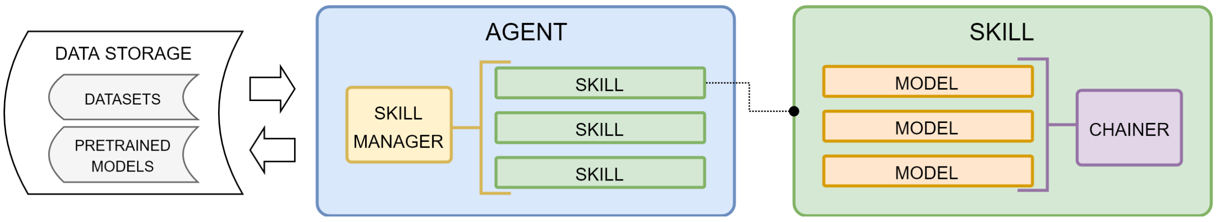

Figure 1: Conceptual architecture of DeepPavlov library.

4.2 Extension of the library

User can easily extend DeepPavlov library by registering a new `Model` or `Skill`. In order to include a new `Model`, a developer should implement a number of standard classes which are used to communicate with the environment:

- **dataset reader** — reads data and returns it in a specified format,
- **dataset iterator** — partitions data into training, validation and test sets, divides the data into batches,
- **vocabulary** — performs data indexing, e.g. converts words into indexes,
- **model** — performs training.

The library contains base classes which implement these functions (`DatasetReader`, `DatasetIterator`, `Vocab` classes). Developers can use them or write their own classes inherited from these base classes. Class for a model can be inherited from an abstract class `NNModel` if it is a neural network, or from a class `Estimator` if it is a non-neural ML model. In addition to that, a user should define a pipeline for the model.

5 Implemented Models and Skills

The library is currently being actively developed with a large set of `Models` and `Skills` already implemented. Some of them are available for interactive online testing.[6]

Skill: Goal-Oriented Dialogue System. The skill implements Hybrid Code Networks (HCNs) described in (Williams et al., 2017). It allows predicting responses in goal-oriented dialogue. The model is configurable: embeddings, slot filling component and intent classifier can be switched on and off on demand. Table 2 shows the performance of our goal-oriented bot on DSTC2 dataset (Henderson et al., 2014). The results demonstrate that our system is close to the state-of-the-art performance.

Model	Test accuracy
Bordes and Weston (2016)	41.1%
Perez and Liu (2016)	48.7%
Eric and Manning (2017)	48.0%
Williams et al. (2017)	55.6%
Deeppavlov*	55.0%

Table 2: Accuracy of predicting bot answers on DSTC2 dataset. *Figures cannot be compared directly, because DeepPavlov model used a different train/test data partition of the dataset.

Model: Entity Recognition. This model is based on BiLSTM-CRF architecture described in (Anh et al., 2017). It is also used for the slot-filling component of the library. Here fuzzy Levenshtein search is used on the recognition results, since the incoming utterances could be noisy. In addition to that, we provide pre-trained NER models for Russian and English. The performance of entity recognition on OntoNotes 5.0 dataset[7] is given in table 3. It shows that our implementation is on par with best-performing models.

Model	F_1-score
DeepPavlov	**87.07** $\pm$ 0.21
Strubell at al. (2017)	86.84 $\pm$ 0.19
Spacy	85.85
Chiu and Nichols (2015)	86.19 $\pm$ 0.25
Durrett and Klein (2014)	84.04

Table 3: Performance of DeepPavlov NER module on OntoNotes 5.0 dataset. Average F_1-score for 18 classes.

Model: Intent Classification. The model implements neural network architecture based on shallow-and-wide Convolutional Neural Network

[6]`http://demo.ipavlov.ai`

[7]`https://catalog.ldc.upenn.edu/ldc2013t19`

(Kim, 2014) and allows multi-label classification of sentences. We do benchmarking for this model on SNIPS dataset[8] and compare its performance with a number of available NLP services. The results given in the table 4 show that our intent classification model is comparable with other existing solutions.

Model	F_1-score
DeepPavlov	**99.10**
api.ai	98.68
IBM Watson	98.63
Microsoft LUIS	98.53
Wit.ai	97.97
Snips.ai	97.87
Recast.ai	97.64
Amazon Lex	97.59

Table 4: Performance of DeepPavlov intent recognition on SNIPS dataset. Average F_1-score for 7 categories. All scores except DeepPavlov are from Inten.to study[10].

Model: Spelling Correction. The component is based on work (Brill and Moore, 2000) and uses statistics-based error model, a static dictionary and an ARPA language model (Paul and Baker, 1992) to correct spelling errors. We tested it on the dataset released for SpellRuEval[11] — a competition on spelling correction for Russian. In table 5 we compare its performance with Yandex.Speller[12] service and open-source spellchecker GNU Aspell[13]. Our model is worse than Yandex.Speller, but it is better then Aspell which is the only freely available spelling correction tool. Even our baseline model outperforms Aspell by large margin, and use of a language model further boosts its performance.

Other Models. The library also contains a sequence-to-sequence goal-oriented bot, and a model for ranking texts by similarity. There are also models which are currently being developed and prepared for publication.

[8]https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines/

[11]http://www.dialog-21.ru/en/evaluation/2016/spelling_correction/

[12]https://tech.yandex.ru/speller/

[13]http://aspell.net/

Method	Precision	Recall	F-score
Yandex.Speller	83.09	59.86	69.59
DeepPavlov	41.42	37.21	39.20
DeepPavlov + LM	51.92	53.94	52.91
GNU Aspell	27.85	34.07	30.65

Table 5: Performance of DeepPavlov spellchecker for Russian.

6 Conclusion

DeepPavlov is an open-source library for developing dialogue agents in Python. It allows assembling a dialogue system from building blocks that implement models for required NLP functionality. These blocks can be recombined and reused in agents for different dialogue tasks. Such modularity opens possibilities for fast prototyping and knowledge transfer. The library supports creation of multi-purpose agents with diverse `Skills`. This is important for real life application scenarios because skills can be added, upgraded or removed independently when a dialogue system is already deployed. New products and conversational solutions can utilise existing skills for faster development. The library currently contains a range of `Models` for solving various NLP tasks, as well as three `Skills`: two goal-oriented and a question-answering one. The library can be easily extended with new `Models` and `Skills`.

DeepPavlov is now being actively developed, and there are many directions for future work. Implementation of models for `Skill Manager` to enable developers assemble full-fledged dialogue agents with a possibility to switch between multiple `Skills` is one of the priorities. Other changes are in progress to improve the usability of the library. Users will be able to define the pipeline directly in the code bypassing a JSON config file. Furthermore, data loading and reading will be simplified, i.e. the majority of datasets will be loaded by a universal data reader.

Finally, we are working on publishing pre-trained models for Russian. This often involves not only training of a model on a Russian dataset, but sometimes changing the model itself.

We recognise that collaboration is an essential part of any scientific, technological and open-source project. DeepPavlov is open to comments, bug reports, feature requests and contributions to our GitHub repo.

Acknowledgements

The work was supported by National Technology Initiative and PAO Sberbank project ID 0000000007417F630002.

References

Le Tanh Anh, Mikhail Y Arkhipov, and Mikhail S Burtsev. 2017. Application of a hybrid bi-lstm-crf model to the task of russian named entity recognition. *arXiv preprint arXiv:1709.09686* .

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR* abs/1605.07683. http://arxiv.org/abs/1605.07683.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 286–293.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *CoRR* abs/1711.01731. http://arxiv.org/abs/1711.01731.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308* .

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics* 2:477–490.

Mihail Eric and Christopher D. Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *CoRR* abs/1701.04024. http://arxiv.org/abs/1701.04024.

M. Gardner, J Grus, M Neumann, O. Tafjord, P. Dasigi, N. Liu, M Peters, M. Schmitz, and L. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform Accessed: 2018-03-23.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Lihong Li, Bhuwan Dhingra, Jianfeng Gao, Xiujun Li, Yun-Nung Chen, Li Deng, and Faisal Ahmed. 2018. End-to-end learning of dialogue agents for information access.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476* .

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch .

Douglas B Paul and Janet M Baker. 1992. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, pages 357–362.

Julien Perez and Fei Liu. 2016. Gated end-to-end memory networks. *CoRR* abs/1610.04211. http://arxiv.org/abs/1610.04211.

R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J.P. Robichaud, A. Celikyilmaz, Y.B. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T. Anastasakos, Z. Ramesh N. Feizollahi, H. Suzuki, R. Holenstein, and E. Radostev V. Krawczyk. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *IEEE Workshop on Spoken Language Technology*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2670–2680.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* .