**DeepPavlov**

# Conceptual overview

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

* A set of pre-trained NLP models, pre-defined dialog system components (ML/DL/Rule-based), and pipeline templates;
* A framework for implementing and testing their own dialog models;
* Tools for application integration with adjacent infrastructure (messengers, helpdesk software, etc.);
* Benchmarking environments for conversational models and uniform access to relevant datasets.

## Key Concepts

The smallest building block of the library is a Component. A Component 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.

Components can be joined into a Model or a Skill. A Model solves a larger NLP task than a Component. However, in terms of implementation, Models are not different from Components. What differentiates a Skill from a Model is that the input and output of a Skill should both be strings. Therefore, Skills are usually associated with dialogue tasks.

DeepPavlov is built on top of the machine learning frameworks [TensorFlow](https://www.tensorflow.org/), [Keras](https://keras.io/) and [PyTorch](https://www.pytorch.org/). Other external libraries can be used to build basic components.

# Choose the Framework

DeepPavlov is built on top of the machine learning frameworks [TensorFlow](https://www.tensorflow.org/), [Keras](https://keras.io/) and [PyTorch](https://www.pytorch.org/):

* BERT-based models on TensorFlow and PyTorch;
* Text classification on Keras and PyTorch;
* Text ranking and morpho-tagging on Keras;
* All other models on TensorFlow.

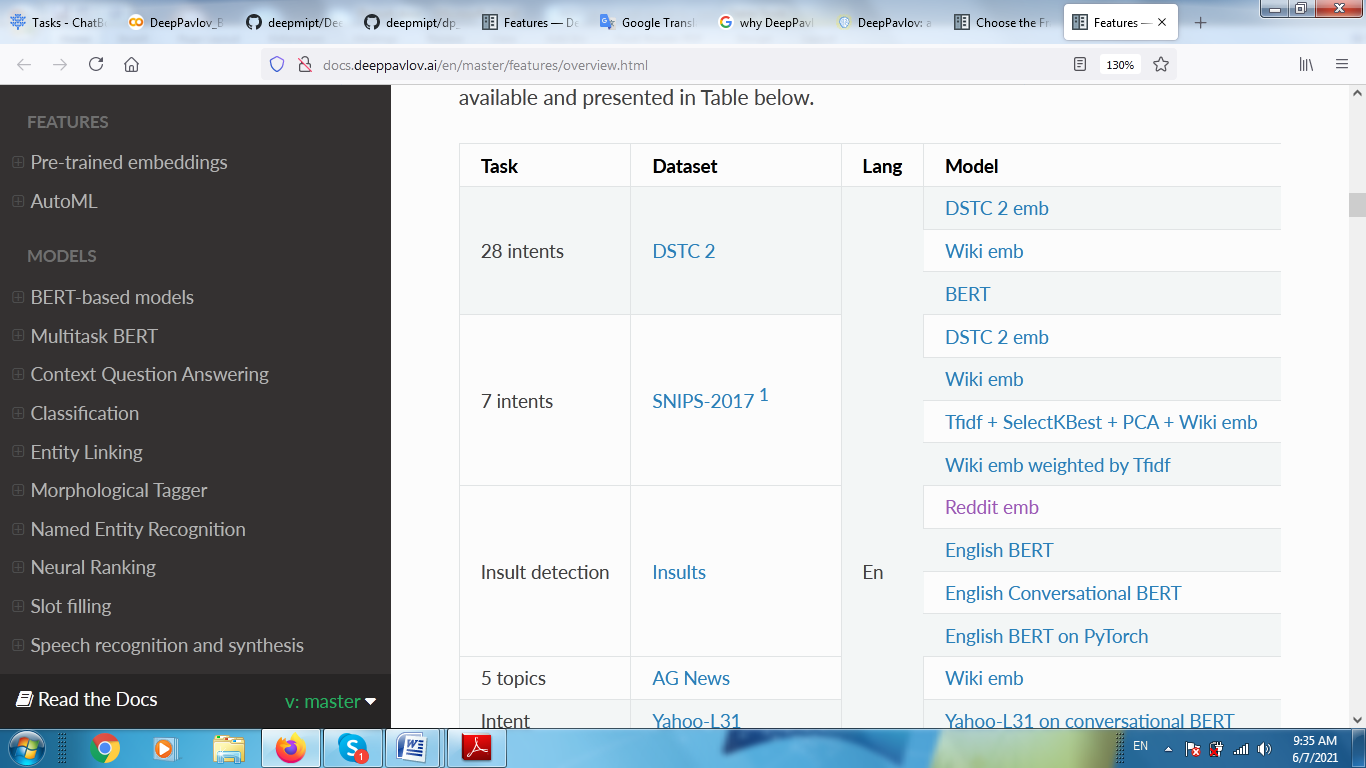
Depending on the considered NLP task, you need to choose one of the available frameworks.

# Features

* [Models](http://docs.deeppavlov.ai/en/master/features/overview.html#models)
  + [NER model](http://docs.deeppavlov.ai/en/master/features/overview.html#ner-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/ner.html)
  + [Slot filling models](http://docs.deeppavlov.ai/en/master/features/overview.html#slot-filling-models-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/slot_filling.html)
  + [Classification model](http://docs.deeppavlov.ai/en/master/features/overview.html#classification-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/classifiers.html)
  + [Automatic spelling correction model](http://docs.deeppavlov.ai/en/master/features/overview.html#automatic-spelling-correction-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/spelling_correction.html)
  + [Ranking model](http://docs.deeppavlov.ai/en/master/features/overview.html#ranking-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/neural_ranking.html)
  + [TF-IDF Ranker model](http://docs.deeppavlov.ai/en/master/features/overview.html#tf-idf-ranker-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/tfidf_ranking.html)
  + [Question Answering model](http://docs.deeppavlov.ai/en/master/features/overview.html#question-answering-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/squad.html)
  + [Morphological tagging model](http://docs.deeppavlov.ai/en/master/features/overview.html#morphological-tagging-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/morphotagger.html)
  + [Syntactic parsing model](http://docs.deeppavlov.ai/en/master/features/overview.html#syntactic-parsing-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/models/syntaxparser.html)
  + [Frequently Asked Questions (FAQ) model](http://docs.deeppavlov.ai/en/master/features/overview.html#frequently-asked-questions-faq-model-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/skills/faq.html)
* [Skills](http://docs.deeppavlov.ai/en/master/features/overview.html#skills)
  + [Goal-oriented bot](http://docs.deeppavlov.ai/en/master/features/overview.html#goal-oriented-bot-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/skills/go_bot.html)
  + [ODQA](http://docs.deeppavlov.ai/en/master/features/overview.html#odqa-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/skills/odqa.html)
* [AutoML](http://docs.deeppavlov.ai/en/master/features/overview.html#automl)
  + [Hyperparameters optimization](http://docs.deeppavlov.ai/en/master/features/overview.html#hyperparameters-optimization-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/hypersearch.html)
* [Embeddings](http://docs.deeppavlov.ai/en/master/features/overview.html#embeddings)
  + [Pre-trained embeddings](http://docs.deeppavlov.ai/en/master/features/overview.html#pre-trained-embeddings-docs) [[docs]](http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html)
* [Examples of some models](http://docs.deeppavlov.ai/en/master/features/overview.html#examples-of-some-models)

### [Classification model](http://docs.deeppavlov.ai/en/master/features/overview.html#id19)

Model for classification tasks (intents, sentiment, etc) on word-level. Shallow-and-wide CNN, Deep CNN, BiLSTM, BiLSTM with self-attention and other models are presented. The model also allows multilabel classification of texts. Several pre-trained models are available and presented in Table below.



### Other Pretrain models:

### [Automatic spelling correction model](http://docs.deeppavlov.ai/en/master/features/overview.html#id20)

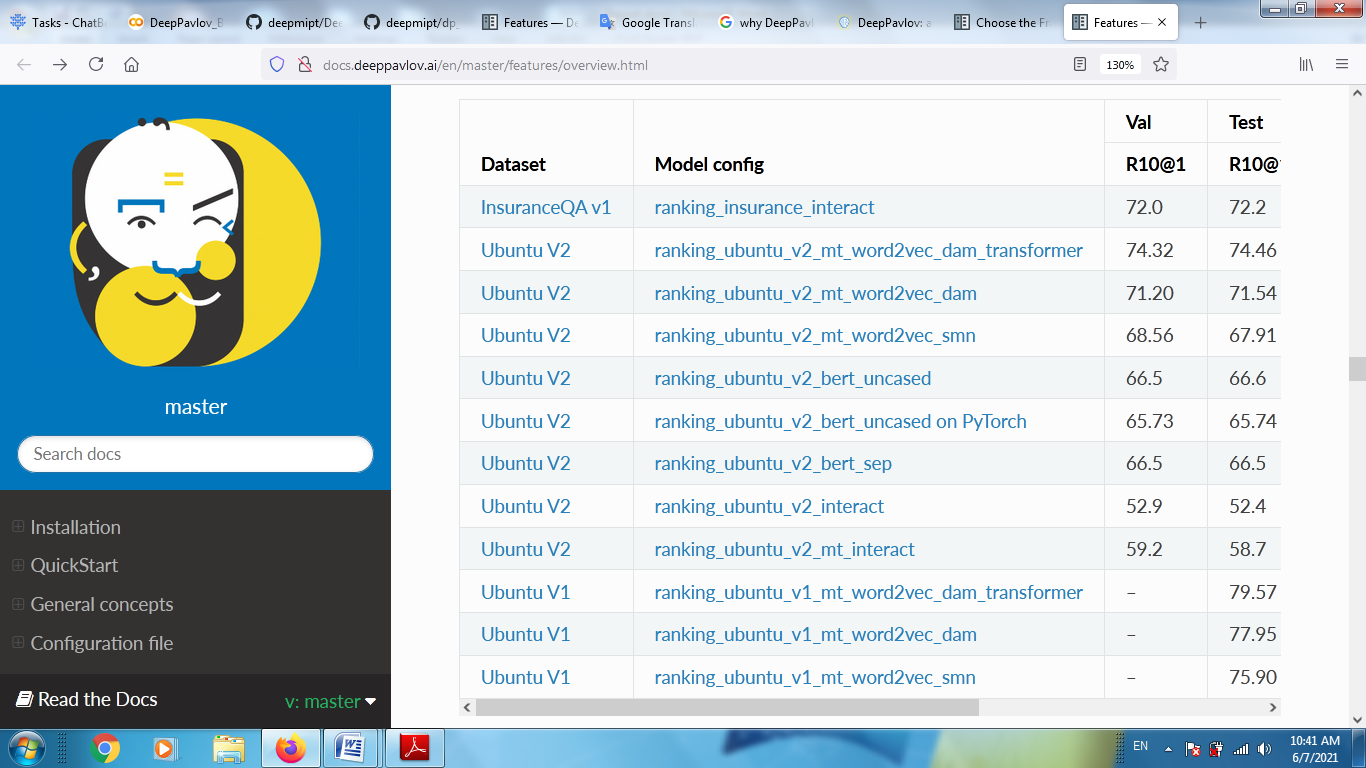
* Available pre-trained models for ranking
* Available pre-trained models for paraphrase identification
* TF-IDF Ranker model
* Question Answering model
* Morphological tagging model
* Syntactic parsing model
* Frequency Asked Questions (FAQ) model

### [Automatic spelling correction model](http://docs.deeppavlov.ai/en/master/features/overview.html#id20)

Pipelines that use candidates search in a static dictionary and an ARPA language model to correct spelling errors.

### [Ranking model](http://docs.deeppavlov.ai/en/master/features/overview.html#id21)

The model performs ranking of responses or contexts from some database by their relevance for the given context.



### [TF-IDF Ranker model](http://docs.deeppavlov.ai/en/master/features/overview.html#id22)

Based on [Reading Wikipedia to Answer Open-Domain Questions](https://github.com/facebookresearch/DrQA/). The model solves the task of document retrieval for a given query.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Model | Wiki dump | Recall@5 | Downloads |
| [SQuAD-v1.1](https://arxiv.org/abs/1606.05250) | [doc\_retrieval](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/doc_retrieval/en_ranker_tfidf_wiki.json) | enwiki (2018-02-11) | 75.6 | 33 GB |

### [Question Answering model](http://docs.deeppavlov.ai/en/master/features/overview.html#id23)

Models in this section solve the task of looking for an answer on a question in a given context ([SQuAD](https://rajpurkar.github.io/SQuAD-explorer/) task format). There are two models for this task in DeepPavlov: BERT-based and R-Net. Both models predict answer start and end position in a given context.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Model config | lang | Downloads |
| [SQuAD-v1.1](https://arxiv.org/abs/1606.05250) | [DeepPavlov BERT](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_bert.json) | en | 806Mb |
| [SQuAD-v1.1](https://arxiv.org/abs/1606.05250) | [DeepPavlov BERT on PyTorch](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_torch_bert.json) | en | 1.1 Gb |
| [SQuAD-v1.1](https://arxiv.org/abs/1606.05250) | [DeepPavlov R-Net](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad.json) | en | ~2.5Gb |
| SDSJ Task B | [DeepPavlov RuBERT](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_ru_bert_infer.json) | ru | 1325Mb |
| SDSJ Task B | [DeepPavlov multilingual BERT](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_ru_bert_infer.json) | ru | 1323Mb |
| SDSJ Task B | [DeepPavlov R-Net](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_ru.json) | ru | ~5Gb |
| [DRCD](https://arxiv.org/abs/1806.00920) | [DeepPavlov multilingual BERT](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_zh_bert_mult.json) | ch | 630Mb |
| [DRCD](https://arxiv.org/abs/1806.00920) | [DeepPavlov Chinese BERT](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/squad/squad_zh_bert_zh.json) | ch | 362Mb |

### [Morphological tagging model](http://docs.deeppavlov.ai/en/master/features/overview.html#id24)

We have a BERT-based model for Russian and character-based models for 11 languages. It is a state-of-the-art model for Russian and near state of the art for several other languages. Model takes as input tokenized sentences and outputs the corresponding sequence of morphological labels in [UD format](http://universaldependencies.org/format.html). The table below contains word and sentence accuracy on UD2.0 datasets. Morphological tagging is the task of assigning labels to a sequence of tokens that describe them morphologically. As compared to Part-of-speech tagging, morphological tagging also considers morphological features, such as case, gender or the tense of verbs.

### [Syntactic parsing model](http://docs.deeppavlov.ai/en/master/features/overview.html#id25)

We have a biaffine model for syntactic parsing based on RuBERT. It achieves the highest known labeled attachments score of 93.7% on ru\_syntagrus Russian corpus (version UD 2.3).

### [Frequently Asked Questions (FAQ) model](http://docs.deeppavlov.ai/en/master/features/overview.html#id26)

Set of pipelines for FAQ task: classifying incoming question into set of known questions and return prepared answer. You can build different pipelines based on: tf-idf, weighted fasttext, cosine similarity, logistic regression.

## [Embeddings](http://docs.deeppavlov.ai/en/master/features/overview.html#id32)

### [Pre-trained embeddings](http://docs.deeppavlov.ai/en/master/features/overview.html#id33)

Word vectors for the Russian language trained on joint [Russian Wikipedia](https://ru.wikipedia.org/) and [Lenta.ru](https://lenta.ru/) corpora.

## [Examples of some models](http://docs.deeppavlov.ai/en/master/features/overview.html#id34)

* Run goal-oriented bot with Telegram interface:

python -m deeppavlov telegram gobot\_dstc2 -d -t <TELEGRAM\_TOKEN>

* Run goal-oriented bot with console interface:

python -m deeppavlov interact gobot\_dstc2 -d

* Run goal-oriented bot with REST API:

python -m deeppavlov riseapi gobot\_dstc2 -d

* Run slot-filling model with Telegram interface:

python -m deeppavlov telegram slotfill\_dstc2 -d -t <TELEGRAM\_TOKEN>

* Run slot-filling model with console interface:

python -m deeppavlov interact slotfill\_dstc2 -d

* Run slot-filling model with REST API:

python -m deeppavlov riseapi slotfill\_dstc2 -d

**Pre-trained embeddings**

* BERT
* ELMO
* FastText

BERT Embedding:

We are publishing several pre-trained BERT models:

* RuBERT for Russian language
* Slavic BERT for Bulgarian, Czech, Polish, and Russian
* Conversational BERT for informal English
* Conversational BERT for informal Russian
* Sentence Multilingual BERT for encoding sentences in 101 languages
* Sentence RuBERT for encoding sentences in Russian

**ELMO**:

We are publishing [Russian language ELMo embeddings model](http://docs.deeppavlov.ai/en/master/apiref/models/embedders.html#deeppavlov.models.embedders.elmo_embedder.ELMoEmbedder) for tensorflow-hub and [LM model](http://docs.deeppavlov.ai/en/master/apiref/models/elmo.html#deeppavlov.models.elmo.elmo.ELMo) for training and fine-tuning ELMo as LM model.

ELMo (Embeddings from Language Models) representations are pre-trained contextual representations from large-scale bidirectional language models. See a paper [Deep contextualized word representations](https://arxiv.org/abs/1802.05365) for more information about the algorithm and a detailed analysis.

## fastText:

We are publishing pre-trained word vectors for Russian language. Several models were trained on joint [Russian Wikipedia](https://ru.wikipedia.org/) and [Lenta.ru](https://lenta.ru/) corpora. We also introduce one model for Russian conversational language that was trained on [Russian Twitter](https://twitter.com/) corpus. All vectors are 300-dimensional. We used fastText skip-gram (see [Bojanowski et al. (2016)](https://arxiv.org/abs/1607.04606)) for vectors training as well as various preprocessing options (see below). You can get vectors either in binary or in text (vec) formats both for fastText and GloVe.

**BERT in DeepPavlov**

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer pre-trained on masked language model and next sentence prediction tasks. This approach showed state-of-the-art results on a wide range of NLP tasks in English. There are several pre-trained BERT models released by Google Research, more details about these pre-trained models could be found here: <https://github.com/google-research/bert#pre-trained-models>.

The deeppavlov\_pytorch models are designed to be run with the [HuggingFace’s Transformers](https://huggingface.co/transformers/) library.

Additionaly the embeddings can be easily used in DeepPavlov. To get text level, token level and subtoken level representations, you can use or modify a [BERT embedder configuration](https://github.com/deepmipt/DeepPavlov/blob/0.15.0/deeppavlov/configs/embedder/bert_embedder.json):

* BERT as Embedder
* Bert for Classificaition
* BERT for Named Entity Recognition (Sequence Tagging)
* BERT for Morphologhical Tagging
* BERT for Syntactic Parsing
* BERT for Context Question Answering (SQUAD)
* BERT for Ranking
* Using custom BERT in DeepPavlov

## BERT for Classification

Two main components of BERT classifier pipeline in DeepPavlov are [BertPreprocessor](http://docs.deeppavlov.ai/en/master/apiref/models/bert.html#deeppavlov.models.preprocessors.bert_preprocessor.BertPreprocessor) on TensorFlow ([TorchTransformersPreprocessor](http://docs.deeppavlov.ai/en/master/apiref/models/torch_bert.html#deeppavlov.models.preprocessors.torch_transformers_preprocessor.TorchTransformersPreprocessor) on PyTorch) and [BertClassifierModel](http://docs.deeppavlov.ai/en/master/apiref/models/bert.html#deeppavlov.models.bert.bert_classifier.BertClassifierModel) on TensorFlow ([TorchTransformersClassifierModel](http://docs.deeppavlov.ai/en/master/apiref/models/torch_bert.html#deeppavlov.models.torch_bert.torch_transformers_classifier.TorchTransformersClassifierModel) on PyTorch).

bert\_classifier and torch\_bert\_classifier have a dense layer of number of classes size upon pooled outputs of Transformer encoder, it is followed by softmax activation (sigmoid if multilabel parameter is set to true in config).

**Multi-task BERT in DeepPavlov**

Multi-task BERT in DeepPavlov is an implementation of BERT training algorithm published in the paper “Multi-Task Deep Neural Networks for Natural Language Understanding”.

The idea is to share BERT body between several tasks. This is necessary if a model pipe has several components using BERT and the amount of GPU memory is limited. Each task has its own ‘head’ part attached to the output of the BERT encoder. If multi-task BERT has *T* heads, one training iteration consists of

* composing *T*  mini-batches, one for each task,

 *T* gradient steps, one gradient step for each task.

When one of BERT heads is being trained, other heads’ parameters do not change. On each training step both BERT head and body parameters are modified. You may specify different learning rates for a head and a body.

Currently there are heads for classification (mt\_bert\_classification\_task) and sequence tagging (mt\_bert\_seq\_tagging\_task).

**Context Question Answering:**

**Question Answering Model for SQuAD dataset**

## Task definition

Question Answering on SQuAD dataset is a task to find an answer on question in a given context (e.g, paragraph from Wikipedia), where the answer to each question is a segment of the context.

## Models

There are two models for this task in DeepPavlov: BERT-based and R-Net. Both models predict answer start and end position in a given context. BERT for SQuAD model documentation on TensorFlow [BertSQuADModel](http://docs.deeppavlov.ai/en/master/apiref/models/bert.html#deeppavlov.models.bert.bert_squad.BertSQuADModel) and on PyTorch [TorchBertSQuADModel](http://docs.deeppavlov.ai/en/master/apiref/models/torch_bert.html#deeppavlov.models.torch_bert.torch_bert_squad.TorchBertSQuADModel).

R-Net : Question Answering Model is based on R-Net.

R-Net for SQuAD model documentation: [SquadModel](http://docs.deeppavlov.ai/en/master/apiref/models/squad.html#deeppavlov.models.squad.squad.SquadModel)

## Model usage from Python

## Model usage from CLI

### Interact mode

Model will ask you to type in context and question.

## Pretrained models:

### SquAD and SQuAD with contexts without correct answers

* **SDSJ**
* **DRCD**

**Classification models in DeepPavlov**

List of available classifiers :

* **BERT classifier** builds BERT architecture for classification problem on **TensorFlow** or on **PyTorch**.
* **Keras classifier** builds neural network on Keras with tensorflow backend.
* **PyTorch classifier** builds neural network on PyTorch.
* **Sklearn classifier** builds most of sklearn classifiers.

## BERT models

Two main components of BERT classifier pipeline in DeepPavlov are deeppavlov.models.preprocessors.bert\_preprocessor.BertPreprocessor on TensorFlow (or deeppavlov.models.preprocessors.torch\_transformers\_preprocessor.TorchTransformersPreprocessor on PyTorch) and deeppavlov.models.bert.bert\_classifier.BertClassifierModel on TensorFlow (or deeppavlov.models.torch\_bert.torch\_transformers\_classifier.TorchTransformersClassifierModel on PyTorch). The deeppavlov.models.torch\_bert.torch\_transformers\_classifier.TorchTransformersClassifierModel class supports any Transformer-based model.

## Neural Networks on Keras

**deeppavlov.models.classifiers.KerasClassificationModel** contains a number of different neural network configurations for classification task.

## Pre-trained models

We also provide with **pre-trained models** for classification on DSTC 2 dataset, SNIPS dataset, “AG News” dataset, “Detecting Insults in Social Commentary”, Twitter sentiment in Russian dataset.

[DSTC 2 dataset](http://camdial.org/~mh521/dstc/) does not initially contain information about **intents**, therefore, Dstc2IntentsDatasetIterator (deeppavlov/dataset\_iterators/dstc2\_intents\_interator.py) instance extracts artificial intents for each user reply using information from acts and slots.

**REST API**

Each DeepPavlov model can be easily made available for inference as a REST web service.

**Integration :**

* **Telegram integration**
* **Yandex Alice integration**
* **Amazon Alexa integration**
* **Microsoft Bot Framework integration**

**Telegram integration**

Any model specified by a DeepPavlov config can be launched as a Telegram bot. You can do it using command line interface or using python.

**Yandex Alice integration**

Any model specified by a DeepPavlov config can be launched as a skill for Yandex.Alice. You can do it using command line interface or using python.

**Amazon Alexa integration**

DeepPavlov models can be made available for inference via Amazon Alexa. Because of Alexa predominantly conversational nature (raw text in, raw text out), the best results can be achieved with models with raw text both in input and output (ODQA, SQuAD, etc.).

**Microsoft Bot Framework integration**

Each library model or skill can be made available for inference via Microsoft Bot Framework.

The whole process takes two main steps:

1. Web App Bot setup in Microsoft Azure
2. DeepPavlov skill/model REST service mounting