CHAPTER 26

Deep Learning for the Russian Language

Ekaterina Artemova

26.1 Introduction

Deep learning has conquered the natural language processing (NLP) research area in the mid-2010s. Most research publications were focused on English and showed a significant improvement of results on major datasets. However, languages other than English were out of the scope of early deep learning research. Russian-oriented research first appeared on Russian local venues, such as Dialogue, Artificial Intelligence and Natural Language (AINL) and Analysis of Images, Social networks, and Texts (AIST). Early papers addressed such tasks as text classification and part of speech tagging. As of the late 2010s, a new trend for multilingual model development was established, which resulted in quite a few models for Russian, released by non-Russian universities and technology companies, such as Google or Facebook.

The deep learning breakthrough is grounded on the efficient use of large amounts of data, without any handcrafted features. While traditional statistical-based machine learning algorithms require a lot of manual annotation of textual data, the deep learning methods discover hidden patterns in the data without human help. Before the deep learning era, an NLP practitioner had to manually set hundreds of features: starting from such surface features as "is a word capitalized," or "is there a comma before the word," up to complex features that try to encode semantics. This resulted, among other things, in creating linguistic corpora, such as Russian National Corpus (http://www.ruscorpora.ru/) and OpenCorpora (http://opencorpora.org) (for more, see Chap. 17).

E. Artemova (✉)
Higher School of Economics (HSE University), Saint Petersburg, Russia
e-mail: echernyak@hse.ru

D. Gritsenko et al. (eds.), *The Palgrave Handbook of Digital Russia Studies*, https://doi.org/10.1007/978-3-030-42855-6_26

The advantages of the deep learning approach to text processing are twofold: first, it produces efficient word and sentence representations, sometimes addressed as word and sentence embeddings, which are capable of modeling lexical and grammatical meaning; second, due to multiple nonlinear transformations applied to word and sentence representations inside the deep model, language patterns are learned from actual observations, rather than from human annotations.

Although deep learning treats data differently from traditional machine learning, **training a model** is core to both approaches. The "black box" is a common metaphor to describe what a model is. We can treat any traditional machine learning or deep learning as a black box, which inputs some observations and outputs target labels. For example, for the task of sentiment analysis, the inputs are the users review and the outputs are either "positive" or "negative" labels (for more on sentiment analysis, see Chap. 28). Inside the black box are mathematical functions and objects that have many settings. The model is developed in two stages. During the first stage, which is addressed as the training stage, the model is **trained** to make correct predictions. The model is presented both with the inputs and correct labels and the settings of the model are adjusted so that the model is capable to produce correct answers. The correct labels help to rule the behavior of the model: if the predictions of the model are correct, it is encouraged to behave the same way, otherwise it is punished for incorrect predictions. It is common to say that the model is **supervised** while receiving feedback from correct labels. During the second stage, **prediction** or **inference** stage, the model is only used for prediction and the settings of the model are unchanged.

The procedure of training a model can be compared to the learning-by-doing, educational approach. The model is not presented with any theoretical statements, but rather is trained to perform in an expected way. While traditional machine learning exploits a variety of different models, deep learning apparatus is based on a single notion of artificial neural network, which is loosely inspired by the human brain. The usage of neural networks allows to develop more versatile models, as different types of neural networks are used as building blocks for specific tasks. This makes the models more reusable and easier to adjust to new tasks. Together, the ability to generalize well along with versatility turns deep learning into a powerful framework that is appealing for use in NLP, as it allows to attain a very high performance across many different NLP tasks.

This chapter provides an overview of deep learning applied to Russian NLP. The remainder is organized as follows: Sect. 26.2 introduces the main deep learning architectures, that is, neural network building blocks. Section 26.3 presents a few NLP tasks and Russian-language examples along with the lists of available datasets and models. Section 26.4 concludes.

26.2 Deep Learning Architecture Overview

The process of designing a neural network is similar to cooking a layered cake. An NLP practitioner first thinks of a preliminary sketch of the model and understands what the input to the model is, and what the model should output. Next, the layers are added one by one to the model. The lowest layer is responsible for reading the textual input and creating an efficient representation of the input. The upper layers are aimed at solving the task under consideration and preparing the desired output. The middle, or the hidden, layers do most of the work: hidden language patterns are discovered here by applying numerous nonlinear transformations.

Neural network architectures are constructed from various types of building blocks or layers. A crucial component of neural networks is the embedding layer. It maps words to vectors in a low dimensional space. These vectors, referred to as word embeddings, can be manipulated as any mathematical object: not only is it possible to calculate a similarity between them, but also to sum them up or to subtract them. The closer the words are by lexical meaning, the closer the corresponding word embeddings should be. The construction of word vectors can be treated either as a standalone task (see Sect. 26.3.1 of this chapter) or as a part of the whole neural network training. Word embeddings can be seen as a broad understanding of the grammar and semantics. When pretrained on a large general corpus, such as Wikipedia, word embeddings reveal the understanding of general language that can be adopted for a more specific domain. Word embeddings are shallow representations that only incorporate previous training in the input layer of the network. The upper layer of the network still needs to be trained from scratch.

Two major neural network architectures are Feed Forward Networks (FFNs) and recurrent neural networks (RNNs). The main difference between these architectures is in the way these architectures input the textual data.

FFNs treat the input text as a so-called “bag of words,” disregarding grammar and word order and taking only word frequency into account. For example, the sentence “the cat sat on the mat” would be turned into the following tuple: ([the, cat, sat, mat, on], [2, 1, 1, 1, 1]). Although FFNs are capable of combining the words in a meaningful way, it is still a significant disadvantage for languages with free word order, where the word order heavily affects the meaning of the sentence.

The design of RNNs overcomes the disadvantages of FFNs by introducing a built-in memory mechanism that summarizes the input text. RNNs can be seen as a tool which reads the input text sequentially in a word-by-word fashion. As the memory is updated after reading a new word, RNNs are endowed with memorizing the word order and the understanding of the current word context. RNNs are usually treated as the analytical module of the whole network and are rarely used as a standalone component. The power of RNNs is in their ability to produce context-aware word representations, which help, for example, to disambiguate word senses. RNNs often work in tandem with

FFNs, so that the output of the RNN is fed into a FFN, intended for final prediction.

The duality of feedforward and recurrent neural networks is caused by the difference of two widely used models for text representation. In contrast to the bag-of-words model exploited by FFNs, the recurrency targets at language modeling, which is central to the majority of NLP tasks. A language model has a double purpose: first, it assigns a probability to a sequence of words. Second, it predicts the next word based on a number of previously used words. The probability of a sentence, estimated by a language model, is closely related to the quality and correctness of the sentence. Language models help to evaluate the quality of machine translation or any other natural language generation task. By predicting the next word, the language model creates context-dependent word and sentence representations.

Although one of the early works by Bengio et al. (2003) shows that FFN can be treated as a language model, RNN outperforms by far FFNs for the task of language modeling. Finally, technical limitations of vanilla RNNs are resolved by gated architectures, such as long short-term memory (LSTM) and gated recurrent unit (GRU) networks. Both LSTM and GRU are very efficient as language models and are de-facto baseline NLP architectures.

The building blocks of neural network architectures are not limited to feed-forward and recurrent layers. Convolutional neural networks (CNNs) are an extension of the FFN architecture. CNNs excel in discovering local patterns. They can be seen as a magnifier, which moves over a word sequence and identifies important features. CNNs are often utilized on the lowest network layers to process not words, but rather characters, to discover long orthographic and derivational patterns. Many applications in Russian, a morphologically rich language, benefit from the ability of CNNs to capture derivational word suffixes and endings. It helps to handle rare words, such as family names, terminology, toponyms, and slang, as well as to take surface features into account (Fig. 26.1).

When compared to feedforward and convolutional neural networks, recurrent neural networks are much slower to train, since they pose long-term dependencies and it is hard to parallelize recurrent computations. The recently introduced transformer layer combines the best of two approaches. It consists of multiple feedforward layers and a powerful attention mechanism that is analogous to human attention in the same way the artificial neural networks model biological neural networks. The attention mechanism directs focus to a certain part of the task while maintaining a background understanding of the whole task. It models word-by-word interactions on each feedforward layer, so that different types of dependencies are considered. The self-attention mechanism is used both to produce context-aware word embeddings, and also measures how strong the dependencies are between the words.

At the core of the recent paradigm shift in NLP, are pretrained language models that are built with rare exceptions with transformer blocks. Not only word embeddings, but the whole neural network is now pretrained as a language model. It becomes possible, since the language modeling objective, next

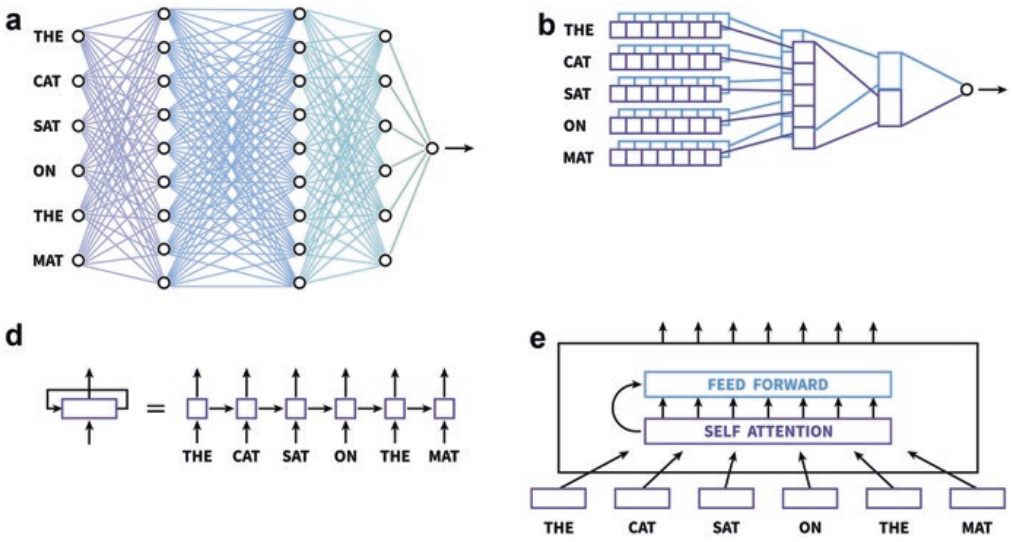

Fig. 26.1 Neural network layers. (**a**) feed forward layer, (**b**) convolutional building layer, (**c**) recurrent layer, (**d**) transformer layer

word prediction, does not require any human annotation. The training data comes for free and the amount of training data available in almost every language are potentially unlimited. Transformer-derived language models seem to capture many facets of language relevant for other NLP tasks. When pretrained on large and diverse corpora, they can be fine-tuned for downstream tasks and surpass previous results in almost every application (for more on corpus linguistics, see Chap. 17).

Despite having excellent results for NLP tasks, neural networks have some disadvantages. First of all, they are frequently treated as black boxes as they lack interpretability. There have been several attempts to find a plausible explanation of how exactly neural networks operate. One of the hypotheses states that the neural network follows the common linguistic pipeline of staged processing of the language. It has been shown that if the neural network is deep enough, lower layers may become morphology aware, middle layers model syntactic dependencies, while the upper layers discover complex semantic patterns. Secondly, deep learning technologies require a lot of data and computational sources. Modern computations, which may take about a month of training, are worth thousands of dollars. Thirdly, ethical concerns arise when training a model on textual data collected from the Web. A model can become unfair when trained on all misconceptions, offensive and biased judgments, fake news and false facts, published on the Web (for more on Runet, see Chap. 16). Finally, the fluency of text generation models may lead to potentially harmful usage. New breed of text generation models impresses with their ability to generate coherent text from minimal prompts. When provided with a headline, such a model will compose a news story; when provided with a movie title, it will compose a movie plot. Text generation models can often

give the appearance of common sense and intelligence, so that it may become quite challenging to recognize, whether a text was composed by a human or by a machine. This frustrates research progress in language generation development, as, it sees, text generators may be misused to generate fake news or propaganda or to increase the amount of spam on the Web. It is of crucial importance, the release of a powerful text generator is accompanied with a tool, which is capable of recognizing machine generated text and can be used to tackle online disinformation.

26.3 NLP Tasks

26.3.1 Word Embeddings: How Do Computers Understand Lexical Meaning

Word embedding stands for a group of methods which are used to map words from a large vocabulary, to vectors. These vectors should consist of real numbers, have few zeros and be of relatively small dimensionality: it is common to construct 300-dimensional word embeddings. These vectors are treated as mathematical objects: not only similarity (or distance) between them can be computed, but also they can be added together or subtracted. At the core of numerous methods for word embedding construction is the distributional hypothesis: words that occur in the same contexts tend to have similar meanings (Harris 1954). Word embedding models are trained on large text corpora. They aim at finding words that share contexts and represent them with such vectors that would be close, according to a mathematical similarity measure. For example, the embeddings of such words as *kofe* ("coffee") and *čaj* ("tee") should have a high similarity degree, since they are used in a similar way, along with the words *pit'* ("to drink"), *čaška* ("cup"), nalit' ("to pour"), et cetera. What is more, advanced word embedding models allow to conduct arithmetical operations: *kofe* ("coffee") to *utro* ("morning") = "*čaj* ("tee") to *večer* ("evening"); *Moskva* ("Moscow") to *Rossiâ* ("Russia") = *Berlin* ("Berlin") to *Germaniâ* ("Germany"). Of course, these associations are corpus-specific and may not be present in other models. The examples are provided by RusVectores (https://rusvectores.org), a free online service which provides, and which computes semantic relations between words in Russian and provides pretrained distributional semantic models (word embeddings), including contextualized ones.

Word embeddings may serve as input to a neural network model, which further will be trained for any downstream task, and may be used as a standalone model for studies of language usage. Word embeddings help to detect semantic shift, caused by either diachronic (Kutuzov et al. 2018) or social changes (Solovyev et al. 2015). Bilingual word embeddings help to develop dictionaries and find similar concepts in different languages (Gordeev et al. 2018).

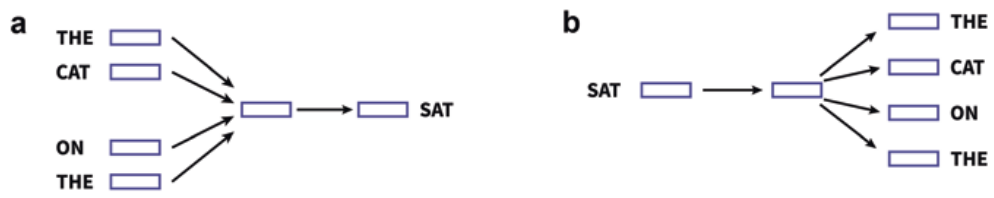

Fig. 26.2 Word2vec configurations. (**a**) continuous bag of words, (**b**) skip-gram

The most popular word embedding model is word2vec (Mikolov et al. 2013) and its extension fasttext (Joulin et al. 2017). Word2vec exploits neural networks to compute word embeddings. It has two configurations: in a continuous bag of words, CBOW, it predicts a word based on surrounding words (two to the left and two to the right). In skip-gram, SGNS, it predicts surrounding words based on the given central word (Fig. 26.2).

SGNS is a de-facto state of the art model for word embeddings and is almost a default choice for many NLP applications for the English language. However, for the Russian language SGNS might not be the best choice. When trained on raw texts, SGNS does not take into account the derivational forms of the words. As a result, for the word *kot* (a cat) there might be up to ten possible vectors for each possible derivational form. This would make a similarity measure almost invalid, since the closest words to the vector *kot* (a cat) would be the vectors of the derivational forms *kotu* (to the cat), *kote* (about the cat), et cetera. To overcome this issue, a preliminary normalization is required to replace each word with its base form. Normalization methods, however, may either have limited vocabulary and introduce some mistakes while processing out of vocabulary words or require word embeddings. This vicious circle is broken by the fasttext model that does not modify word2vec mathematics but treats the words differently. Instead of computing a single vector for a given word, it computes multiple vectors for all character n-grams (sequences of two to five characters) and then combines them to get the final vector.

Fasttext allows to capture such properties of rich morphology in Russian as derivational patterns in suffixes and endings. It is strongly recommended to use fasttext for the Russian language as the word embedding model. See Table 26.1 for available pretrained word embedding models and Table 26.2 for word embedding training tools.

Word embedding models often fail when faced with such complex language phenomena as antonyms or homonyms. Although word embeddings are exceptionally powerful for finding words that share a similar meaning, they often mistake for words that have opposite meanings, such as *proigrat'* ("to lose") or *vyigrat'* ("to win"), as they occur in similar contexts. Word embedding models suffer from polysemy and homonymy. Such words as *luk* ("onion" or "bow" or "a look") and *zamok* ("castle" or "lock") get a single vector, despite having multiple sense. A few models, such as AdaGram (Bartunov et al. 2016) and SenseGram (Pelevina et al. 2016), try to overcome this issue by

Table 26.1 Word embeddings for Russian

http://rusvectores.org	Multiple Russian-only word and sentence embeddings (Kutuzov and Kuzmenko, 2017)
http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html	Multiple Russian-only word and sentence embeddings
https://fasttext.cc/docs/en/crawl-vectors.html	Google fasttext embeddings trained with limited preprocessing for 157 languages
http://vectors.nlpl.eu/repository/	Thirteen Russian word embedding models trained with clearly stated hyperparameters, on clearly described and linguistically preprocessed corpora
https://github.com/bheinzerling/bpemb	BPE embeddings for 275 languages

Table 26.2 Tools to train word embeddings

Library	*Language*	*URL*
Gensim	Python	https://radimrehurek.com/gensim/
AllenNLP	Python	https://github.com/allenai/allennlp
flair	Python	https://github.com/zalandoresearch/flair
fasttext	C++/terminal interface	https://fasttext.cc
Deeplearning4j	Java/Scala	http://deeplearning4j.org

simultaneous word sense disambiguation, and word embedding training. However, current pretrained language models are a much more efficient solution to this issue, as they search for context-dependent word embeddings.

As of the mid 2010s, using pretrained word embeddings as an input to any machine learning or deep learning has become a must. The word embeddings can be fine-tuned while training the model for a downstream task or remain constant. Fine-tuning of word embeddings may help to resolve some issues related to antonyms or homonyms. When fine-tuned for sentiment classification (for more on Sentiment analysis, see Chap. 28), embeddings for words *horošij* ("good") and *plohoj* ("bad"), which may be initially close, will be pushed apart from each other.

Last but not least, an alternative approach to word tokenization, called byte pair encoding (BPE; Heinzerling and Strube 2018), suggests not to use whole words as text units, but rather split the words into subwords, based on frequent n-grams. BPE tokens resemble to a certain degree, morphemes, and seem quite promising for Russian.

To conclude this section, we will list a few pretrained word embedding models for Russian in Table 26.1.

All these models are available for downloads as single files. The models are trained on large freely available corpora, such as Wikipedia, Taiga,[1] and

Araneum.[2] The vocabulary of the models ranges from 100K to 700K unique tokens and the model size ranges from 200MB to 3GB.

RusVectores additionally provides web interface for exploration of word embedding models, along with visualization and semantic calculator.

Table 26.2 lists tools freely available to train embedding models from scratch. Gensim is one of the most popular Python libraries for building word embedding models and topic models, though Gensim does not provide deep learning functionality. In contrast to Gensim, AllenNLP and flair provide reference implementations for deep learning models for NLP, including word2vec and fasttext. These libraries provide tools for processing textual data and share similar functionality, though target different audience. AllenNLP is more advanced and flair is designed as a very simple framework. Both AllenNLP and flair have Python interfaces. Fasttext is available as a console application of the same name. Deeplearning4j is a general deep learning framework that provides scripts for training deep learning models.

26.3.2 Text Classification

The task of text classification is to assign categories to texts. This is a common supervised task: given labeled data (i.e. texts, annotated with class labels), a model should be first trained, and then applied to unlabeled test data.

Text classification is one of the most demanded industrial NLP tasks. Sentiment analysis and information filtering are the most common applications of text classification algorithms. Sentiment analysis is widely used for marketing research. Companies use sentiment classification for product analytics, brand monitoring, customer support, and market research. One of the main information filtering techniques is spam filtering, which exploit classification algorithms to distinguish between spam and ham incoming emails. In general, email categorization is a powerful idea which facilitates the work of an office employee. Other information filtering applications may include identification of trolls, obscene content detection, ad blocking and privacy protection. What is more, hotlines use text classification for language identification.

Virtual personal assistants, such as Apple Siri or Amazon Alexa, are becoming an internal part of our daily lives. They use the whole range of NLP methods, including text classification. Each user utterance is classified according to its intent, according to the desired action of the user (i.e. whether the user meant to launch an application, make a call, write a note, etc.).

The classification of Russian texts is almost no different from English text classification and follows a standard pipeline:

1. Word embeddings are used as an input to the model
2. Multiple hidden CNN- or RNN-derived layers are used for input processing
3. A feed forward layer is used for final prediction.

The labels in the training set, that is, the correct answers, are used for supervision. When presented with correct answers, the model is able to adjust its own parameters so that its predictions become correct.

The quality of the classification task is evaluated according to the ratio of correct predictions and the ratio of erroneous predictions.

There are a few recent Russian-language datasets for text classification:

- A large-scale dataset for sentiment analysis, which consists of texts from social media (Rogers et al. 2018).
- A dataset for sentiment analysis of product reviews on e-commerce sites (Smetanin and Komarov 2019).
- A collected dataset for humor recognition in short stories (Baranova-Bolotova et al. 2019).
- RusIdiolect[3] is a dataset for experimental studies of the idiolect of a native Russian speaker, such as deception detection (Litvinova et al. 2017).
- RusProfiling is a popular dataset for author profiling, including gender identification. Current state of the art results are achieved by Sboev et al. (2018).

These datasets are available to download from the Web. In contrast to major English datasets gathered in Natural Language Toolkit[4] (NLTK), there is no unified application programming interface (API) to access Russian datasets.

Finally, the major component of fasttext (Joulin et al. 2017) functionality is a simple yet strong classification algorithm. It is very fast and easy to use and is strongly recommended as a strong baseline.

Finally, there are a few applications of word embeddings outside linguistic field. For example, (Panicheva and Litvinova 2019) report on using word embeddings to measure speech coherence of patients, affected by schizophrenia. "Semantic coherence" is defined as mean pairwise similarity between words in a sample text, written by a patient. Word embeddings allow to measure semantic coherence, as they provide a simple approach to measure word similarity. The schizophrenia status of a patient along with text samples is provided in RusIdiolect corpus. The findings of Panicheva and Litvinova show that semantic coherence features allow to distinguish between healthy patients and patients, who suffer from schizophrenia. This is comparable to results reported for similar task in English. This research project aims at studying various phenomena present in the schizophrenia and by no means calls to replace traditional medical diagnostics.

26.3.3 Sequence Labeling

The task of sequence labeling is to assign categories to single words. Common examples of a sequence labeling tasks are part-of-speech (POS) tagging or named entity recognition (NER). POS tagging is the task of labeling a word with a corresponding POS tag. NER seeks to identify such named entities as

Table 26.3 Two examples of sequence labeling tasks

	Boris (Boris)	*Pasternak (Pasternak)*	*rodilsâ (was born)*	*v (in)*	*Moskve (Moscow)*
POS tags	PROPN	PROP	VERB	PREP	PROPN
NE tags	Person	Person	O	O	Location

POS tagging (first line), named entity recognition (second line). Each word is assigned with two tags: a POS tag and a named entity tag. If the word is not a named entity, the tag "O" is used

persons, locations, organizations, et cetera, and assign them with a corresponding tag. See Table 26.3 for examples of POS tagging and NER.

Sequence labeling applications range from linguistics tasks, such as POS tagging, which can be treated is a preliminary step for further analysis, up to more complex tasks, such as coreference and gapping resolution. NER, as a sequence labeling task, can be treated as a preliminary step for machine translation. Named entities should be identified and treated differently from regular words for proper translation. When used in Legal Tech or medical applications, NER helps to discover important features, such as legal condition or diseases, used further for decision-making. In Russian realities, Legal Tech applications are very much in demand. This motivates several research groups to develop NER methods for specific domains.

Sequence labeling helps virtual assistants to understand user needs better. While text classification helps to detect user intent, sequence labeling methods are able to fill in slots, that is, to discover specific details, such as what exactly application should be launched or which contact should be addressed. Gapping and coreference are crucial for handling messaging history. Gapping resolutions helps to find omitted predicates in consequent turns, while coreference resolutions helps to connect nouns and names with corresponding pronouns.

RNN and its variations are widely used for sequence labeling tasks due to its ability to process a sequence word by word. We can think of RNN as an attentive reader that reads each word carefully, thinks over the context of the word, and then makes a decision as to what tag to assign. It is worth noting that bidirectional variations of RNN, capable of both left-to-right and right-to-left reading, are suited to model languages with free word order as they maintain both left and right contexts.

The pipeline of the sequence labeling task does not differ significantly from the text classification pipeline:

1. Word embeddings are used as an input to the model. Word embeddings may be extended with convolved character representations, which would take care of derivational patterns.
2. Multiple hidden RNN-derived layers are used for processing input and for producing context-aware word representations.
3. Each word representation is fed into a feed forward layer for final prediction and each word is assigned with a label. Alternatively, another model,

conditional random field (CRF), may be used on top of the recurrent layer to reweight its prediction.

The main difference between text classification and sequence label affects the final layer. When used for text classification, the final layer is applied only once to get one class label. However, for sequence labeling it is applied to each individual word representation from previous layer.

In contrast to text classification task, sequence labeling seems to be more complicated from a linguistic point of view. Tasks, modeled as sequence labeling, are more advanced and range from POS tagging to coreference and gapping resolution.

There are several Russian datasets for the sequence labeling task:

- Universal dependencies[5] project presents four Russian corpora annotated with POS tags
- Persons-1000 (Gareev et al. 2013) and FactRuEval (Starostin et al. 2016) are large-scale datasets for named entity recognition
- AGGR-2019 (Smurov et al. 2019) is a corpus for gapping resolution
- RuCor and AnCor (Toldova et al. 2014) are corpora used for coreference and anaphora resolution
- SberQUAD, a dataset for question answering, treats answer generation as a retrieval of a relevant fragment of text.

26.3.4 Transfer Learning in NLP

Since 2017, NLP field has witnessed the emergence of transfer learning methods and algorithms. Transfer learning stands for the process of training a model on a large-scale dataset to conduct a simple task, such as language modeling. Next, this pretrained model is trained for the second time for more complicated tasks. The transfer learning process is comparable to the way a child is educated. Children acquire the language from their environment, and only in the school they are taught to complete grammar tasks. The same way models gain language understanding while being pretrained and then are supervised for specific tasks.

Transfer learning led to a paradigm shift in NLP. Instead of using every time pretrained word embeddings and training the whole model from scratch, now a pretrained model is fine-tuned for downstream tasks. This requires much less annotated data and leads to superior results simultaneously. Word embeddings were an imperfect way to store language representation, which suffered from language ambiguity. Pretrained models are less prone to polysemy and antonymy and are able to handle multilinguality at the same time.

Despite the fact that transfer learning paradigms leads to superior results in comparison to previous approaches, so far it has not enabled any exceptionally new applications.

Inside transfer learning models are transformer layers (see Fig. 26.1d) that are more advanced from a technical point of view when compared to other layers. The architecture of transfer learning models is sophisticated, enumerates millions of parameters, and take weeks to be pretrained.

Not only transfer learning models established new state of the art for several existing NLP tasks, they also appear to be efficient in new generations of tasks. For example, there is evidence that the tasks that require commonsense understanding can be conducted using transfer learning techniques. This is supported by the idea that excessive pretraining results in a subtle understanding of language patterns.

When pretrained on the corpus of multiple languages or on parallel corpus, transfer learning models become aware of several languages at the same time and can be shared across several languages for the same downstream task. For example, Piskorski et al. (2019) show how NER in four Slavic languages can be approached by a multilingual model.

Even though pretraining of a large model is expensive and time-consuming, new models appear almost every month as of late 2019. Among others, ELMo (Peters et al. 2018) and BERT by Google (Devlin et al. 2019) are the most popular models. BERT's successors, ALBERT, RoBERTa, XLNet, and T5, released by Facebook, Microsoft, and other technology companies, are larger and outperform BERT by far. At the same time, they are heavily criticized for being unaffordable for smaller institutions. Indeed, few universities in Russia have enough resources to train transfer learning models. Table 26.4 lists transfer learning models available for the Russian language. RusVectores poses both word and sentence embeddings model. RusVectores provides not only word embeddings, but also a pretrained ELMo model, which can be treated as sentence embedding models.

Transfer learning models can be exploited as a standalone sentence embedding tool. Sentence embeddings are massively used in those applications, which require modeling of sentence similarity. Consider, for example, the task of finding an answer to a frequently asked question (FAQ). Imagine that the answers to some FAQs are already known, and a user asks a new question. The most similar question to the new one can be found by using an embedding-based similarity measure. With a high chance, the answer to the retrieved question should fit the new question, too.

Table 26.4 Transfer learning models for Russian

http://docs.deeppavlov.ai/en/master/features/models/bert.html	BERT in DeepPavlov (Kuratov and Arkhipov 2019)
http://rusvectores.org	Multiple Russian-only word and sentence embeddings (Kutuzov and Kuzmenko 2017)
https://github.com/vlarine/ruberta	Russian RoBERTa, RuBERTa

Transfer learning models are excellent not only in solving complex downstream tasks, but also in text generation. Some researchers are afraid of the fluency of these models and raise ethical questions of the harmless strategy of releasing the models. When misused, the transfer learning models can generate fake news and offensive utterances and be disturbing. However, these concerns are vivid for English-spoken communities and do not reach Russia so far.

26.4 Conclusion

This chapter discusses the applications of deep learning methods to Natural Language Processing tasks and is particularly oriented at the Russian language. We traced the development of deep learning methods for NLP from early stages of using feed forward networks to recent developments in transfer learning. Two basic text representation models, namely bag-of-words and language models, were presented and related to the duality between convolutional and recurrent neural networks. We have recognized a recent paradigm shift, caused by new advances in architecture design and development of transformer layers. Several analogies between human intelligence and neural networks were drawn. Neural networks aim at resembling human by using artificial neurons and attention mechanisms, and acquiring language from textual data.

With no doubt, deep learning is a leading paradigm in modern language technology. Unfortunately, the Russian language resources do not provide enough resources to exploit deep learning scope fully. The Russian research community is facing a need for both keeping track of worldwide challenges and, if necessary, reapply the methods initially developed for the English language to the Russian language.

The latter requires an increase not only of computational powers, which is rather a financial matter but also of the amounts of annotated data. Recent government decisions and AI-centered strategies seem to provide financial support to the research community that may help to narrow the gap between English and Russian language resources.

Not only is Russian different from English from a linguistic point of view, but also different language technology applications are demanded in Russia and English-spoken countries. The major NLP applications in Russia are related to marketing, e-Government transformation, and call center automation. Whole domains such as Legal Tech, medical NLP, educational NLP still stay out of business focus and are subject to further development. Minority languages currently are not supported by major language technology applications with Yandex.Search (the web search engine and the core product of Yandex) being the only exception.

At the same time, while language technologies become more and more sophisticated, the entry threshold to the NLP field is lowered. Recent advances in programming tools and programming languages made it possible to develop high-level languages, which can be easily comprehended by users with little or

no previous programming experience. Successful implementations of many deep learning architectures have substantially facilitated the development of practical applications. The complexity of deep learning models comes along with the flexibility of fine-tuning and reuse across practical applications. The nearest future will likely witness the transformation of learnable approaches to daily routines.

Notes

1. https://tatianashavrina.github.io/taiga_site/.
2. http://ucts.uniba.sk/aranea_about/.
3. https://rusidiolect.rusprofilinglab.ru, yet not published.
4. https://www.nltk.org.
5. https://universaldependencies.org.

References

Baranova-Bolotova, V., V. Blinov, and P. Braslavski. 2019. Lightning Talk-Humor Recognition in Russian Language. In *Companion Proceedings of the 2019 World Wide Web Conference*, 1268–1269. ACM.

Bartunov, S., D. Kondrashkin, A. Osokin, and D. Vetrov. 2016. Breaking Sticks and Ambiguities with Adaptive Skip-Gram. *Artificial Intelligence and Statistics*: 130–138.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3 (Feb): 1137–1155.

Devlin, J., M.W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1. (Long and Short Papers), 4171–4186.

Gareev, R., M. Tkachenko, V. Solovyev, A. Simanovsky, and V. Ivanov. 2013. Introducing Baselines for Russian Named Entity Recognition. In *International Conference on Intelligent Text Processing and Computational Linguistics*, 329–342. Berlin, Heidelberg: Springer.

Gordeev, Denis, Alexey Rey, and Dmitry Shagarov. 2018. Unsupervised Cross-Lingual Matching of Product Classifications. In *Proceedings of the 23rd Conference of Open Innovations Association FRUCT*, 62. FRUCT Oy.

Harris, Z.S. 1954. Distributional Structure. *Word* 10 (2–3): 146–162.

Heinzerling, B., and M. Strube. 2018. BPEmb: Tokenization-Free Pre-Trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*. LREC-2018.

Joulin, A., E. Grave, P. Bojanowski, and T. Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics 2, Short Papers*, 427–431.

Kuratov, Y., and M. Arkhipov. 2019. Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language. *arXiv preprint arXiv:1905.07213*.

Kutuzov, A., and E. Kuzmenko. 2017. WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models. In *Analysis of Images, Social Networks and Texts*, *AIST 2016. Communications in Computer and Information Science*, ed. D. Ignatov et al., 661. Cham: Springer.

Kutuzov, A., L. Øvrelid, T. Szymanski, and E. Velldal. 2018. Diachronic Word Embeddings and Semantic Shifts: A Survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1384–1397.

Litvinova, Olga, Pavel Seredin, Tatiana Litvinova, and John Lyell. 2017. Deception Detection in Russian Texts. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 43–52.

Mikolov, T., I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

Panicheva, Polina, and Tatiana Litvinova. 2019. Semantic Coherence in Schizophrenia in Russian Written Texts. In *Proceedings of the 25rd Conference of Open Innovations Association FRUCT*, 240. FRUCT.

Pelevina, M., N. Arefiev, C. Biemann, and A. Panchenko. 2016. Making Sense of Word Embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, 174–183.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of NAACL-HLT*, 2227–2237.

Piskorski, J., L. Laskova, M. Marcińczuk, L. Pivovarova, P. Přibáň, J. Steinberger, and R. Yangarber. 2019. The Second Cross-Lingual Challenge on Recognition, Normalization, Classification, and Linking of Named Entities across Slavic Languages. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, 63–74.

Rogers, Anna, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas, and Alex Gribov. 2018. Rusentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian. In *Proceedings of the 27th International Conference on Computational Linguistics*, 755–763.

Sboev, Alexander, Ivan Moloshnikov, Dmitry Gudovskikh, Anton Selivanov, Roman Rybka, and Tatiana Litvinova. 2018. Deep Learning Neural Nets Versus Traditional Machine Learning in Gender Identification of Authors of RusProfiling Texts. *Procedia Computer Science* 123 (2018): 424–431.

Smetanin, S., and M. Komarov. 2019. Sentiment Analysis of Product Reviews in Russian Using Convolutional Neural Networks. In *2019 IEEE 21st Conference on Business Informatics (CBI)*, vol. 1, 482–486. IEEE.

Smurov, I.M., M. Ponomareva, T.O. Shavrina, and K. Droganova. 2019. Agrr-2019: Automatic Gapping Resolution for Russian. *Computational Linguistics and Intellectual Technologies*: 561–575.

Solovyev, Valery D., Vladimir V. Bochkarev, and A.D. Kaveeva. 2015. Variations of Social Psychology of Russian Society in Last 100 Years. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 519–523. IEEE.

Starostin, A.S., V.V. Bocharov, S.V. Alexeeva, A. Bodrova, A.S. Chuchunkov, S.S. Dzhumaev, and M.A. Nikolaeva. 2016. FactRuEval 2016: Evaluation of Named

Entity Recognition and Fact Extraction Systems for Russian. In *Computational Linguistics and Intellectual Technologies. Proceedings of the Annual International Conference Dialogue (2016)*, vol. 15, 702–720.

Toldova, S.J., A. Roytberg, A.A. Ladygina, M.D. Vasilyeva, I.L. Azerkovich, M. Kurzukov, and Y. Grishina. 2014. RU-EVAL-2014: Evaluating Anaphora and Coreference Resolution for Russian. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue"*, 681–694.