Long short-term memory (synopsis)

Sequences

- An inputsequence can be denoted ($x^{(1)}$, $x^{(2)}$, ..., $x^{(T)}$) where each data point $x^{(t)}$is a realvalued vector. Similarly, a target sequence can be denoted ($y^{(1)}$, $y^{(2)}$, ..., $y^{(T)}$).
- Sequences may be of finite or countablyinfinite length.When they are finite, the maximum time index of the sequenceis called T.
- Using temporal terminology, an input sequence consists of data points $x^{(t)}$that arrive in a discrete sequence of time steps indexed by t.A target sequenceconsists of data points $y^{(t)}$.The time-indexed data points may be equally spaced samples from a continuous real-world process.

Neural networks

- Generally,a neural network consists of a set of artificial neurons, commonly referred toas nodes or units, and a set of directed edges between them, which intuitivelyrepresent the synapses in a biological neural network.
- Associated with eachneuron j is an activation function l_j (), which is sometimes called a link function.
- Associated with each edge from node j’to j is a weight $w_{jj'}$denotes the “to-from” weight corresponding tothe directed edge to node j from node j’.
- The value $v_j$of each neuron j is calculated by applying its activation functionto a weighted sum of the values of its input nodes:

$$\boxed{v_j = l_j\left(\sum_{j'} w_{jj'} \cdot v_{j'}\right).}$$

The activation function (подробнее https://lasagne.readthedocs.io/en/latest/modules/nonlinearities.html):

- the sigmoid (z) =1 / (1 + e^z);
- the tanh function (z) = (e^z e^z) / (e^z+ e^z);
- the rectified linear unit (ReLU) whose formula is l_j(z) = max(0, z). This type of unit has been demonstrated to improve the performance of many deep neural networks;

The activation function at the output nodes depends upon the task:

- For multiclass classification with K alternative classes, we apply a softmax nonlinearity in an output layer of K nodes.Multiclass classificationmeans a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multiclass classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.
- For multilabel classification the activation function is simply a point-wise sigmoid.Multilabel classificationassigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.
- For regression we typically have linear output.

Feedforward networks and backpropagation

The input x to a feedforward network is provided by setting the values ofthe lowest layer. Each higher layer is then successively computed until output is generated at the topmost layer y’.Feedforward networks are frequently usedfor supervised learning tasks such as classification and regression. Learning is accomplished by iteratively updating each of the weights to minimize a loss function, L(y’,y), which penalizes the distance between the output y’and thetarget y.

The most successful algorithm for training neural networks is backpropagation (Rumelhart).Backpropagationuses the chain rule to calculate the derivative of the loss function L with respect to each parameter in the network. The weights are then adjusted by gradient descent. Because the loss surface is non-convex, there is no assurance that backpropagation will reach a global minimum.Nowadays, neural networks are usually trained with stochastic gradient descent (SGD) using mini-batches.Псевдокод:FeedForward_NN_SGD.txt

Many variants of SGD are used to accelerate learning:

- AdaGrad;adapts the learning rate by caching the sum of squared gradients with respect to each parameter at each time step. The step size for each feature is multiplied by the inverse of the square root of this cached value. AdaGrad leads to fast convergence on convex error surfaces, but because the cached sum is monotonically increasing, the method has a monotonically decreasing learning rate, which may be undesirable on highly non-convex loss surfaces. Псевдокод:AdaGrad.txt
- AdaDelta;
- RMSprop;RMSprop modifies AdaGrad by introducing a decay factor in the cache, changing the monotonically growing value into a moving

average.

Momentum methods.These methods add to each update a decaying sum of the previous updates. When the momentum parameter is tuned well and the network is initialized well, momentum methods can train deep nets and recurrent nets competitively with more computationally expensive methods like the Hessian free optimizer (http://andrew.gibiansky.com/blog/machine-learning/hessian-free-optimization/).

Reccurent neural networks

Edges that connect adjacent time steps, called recurrent edges, may form cycles, including cycles of length one that are selfconnections from a node to itself across time.

$$\boxed{\boldsymbol{h}^{(t)} = \sigma(W^{\mathrm{hx}}\boldsymbol{x}^{(t)} + W^{\mathrm{hh}}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)}$$

At time t, nodes with recurrent edges receive input from the current data point$x^{(t)}$,hidden node values $h^{(t-1)}$in the network’s previous state:

$$\boxed{\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(W^{\mathrm{yh}}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y).}$$

Training RNN:

Vanishing / exploding gradinets;

- The problems of vanishing and exploding gradients occur when backpropagating errors across many time steps.
- Which of the two phenomena occurs depends on whether the weight of therecurrent edge $|w_{jj}| > 1$ or $|w_{jj}| < 1$ and on the activation function in the hidden node.
- Given a sigmoid activation function, the vanishing gradient problem is more pressing, but with a rectified linear unit max(0,x), it is easier to imagine the exploding gradient.
- Pascanu give a thorough mathematical treatment of the vanishing and exploding gradient problems, characterizing exact conditions under which these problems may occur. Given these conditions, they suggest an approach to training via a regularization term that forces the weights to values where the gradient neither vanishes nor explodes.

Truncated backpropagation through time (TBPTT) (Williams and Zipser);

- With TBPTT, some maximum number of time steps is set along which error can be propagated.
- While TBPTT with a small cutoff can be used to alleviate the exploding gradient problem, it requires that one sacrifice the ability to learn long-range dependencies.

RNN designs:

- Hopfield introduced a family of recurrent neural networks that have pattern recognition capabilities.They are defined by the values of the weights between nodes and the link functions are simple thresholding at zero. In these nets, a pattern is placed in the network by setting the values of the nodes. The network then runs for some time according to its update rules, and eventually another pattern is read out. Hopfield networks are useful for recovering a stored pattern from a corrupted version and are the forerunners of Boltzmann machines and auto-encoders.
- An early architecture for supervised learning on sequences was introducedby Jordan. Thi nnis a feedforward network with a single hidden layer that is extended with special units. Output node values are fed to the special units, which then feed these values to the hidden nodes at the following time step. If the output values are actions, the special units allow the network to remember actions taken at previous time steps.Additionally, the special units in a Jordan network are self-connected. Intuitively, these edges allow sending information across multiple time steps without perturbing the output at each intermediate time step.
- The architecture introduced by Elman is simpler than the earlierJordan architecture. Associated with each unit in the hidden layer is a context unit. Each such unit j’ takes as input the state of the corresponding hidden node j at the previous time step, along an edge of fixed weight $w_{j'j} = 1$. This value then feeds back into the same hidden node j along a standard edge. This architecture is equivalent to a simple RNN in which each hidden node has a single self-connected recurrent edge.
- The idea of fixed-weight recurrent edges that make hidden nodes self-connected is fundamental in subsequent work on LSTM networks (Hochreiter and Schmidhuber).

From feedforward NN to RNN

Standard neural networks have limitations.Most notably, they rely on the assumption of independence among the trainingand test examples. After each example (data point) is processed, the entire stateof the network is lost. If each example is generated independently, this presentsno problem. But if data points are related in time or space, this is unacceptable.Additionally, standard networks generally rely on examples being vectors of fixed length.For example, a model trainedusing a finite-length context window of length 5 could never be trained to answerthe simple question, “what was the data point seen six time steps ago?”Thusit is desirable to extend these powerful learning tools to model data with temporal or sequential structure and varying length inputs and outputs.Recurrent neural networks (RNNs) are connectionist models with the ability toselectively pass information across sequence steps, while processing sequentialdata one element at a time.

The hidden state of RNN at any time step can containinformation from a nearly arbitrarily long context window.This is possible because the number of distinct states that can be represented in a hidden layer ofnodes grows exponentially with the number of nodes in the layer.While the potentialexpressive power of a network grows exponentially with the number of nodes,the complexity of both inference and training grows at most quadratically.

RNNs are not limited to time-based sequences.They have beenused successfully on non-temporal sequence data, including genetic data.

The derivative of the loss function can be calculated with respect to each of the parameters(weights) in the model.Exploding / vanishing gradients problem.RNNs are extremely difficult to train, one of the main reasons being that whenthe error gradients are backpropagated via the chain rule, they tend to groweither very large or very small, effectively precluding the network from training.

May be HMM?

Hidden Markov models (HMMs), which model an observed sequenceas probabilistically dependent upon a sequence of unobserved states.

- the transitiontable capturing the probability of moving between any two time-adjacent statesis of size |S| **2 . Thus, standard operations become infeasible with an HMM whenthe set of possible hidden states grows large;
- each hidden state candepend only on the immediately previous state;

Thus,it is possible to extend aMarkov model to account for a larger context window,this procedure grows the state space exponentially with the size of thewindow, rendering Markov models computationally impractical for modelinglong-range dependencies.

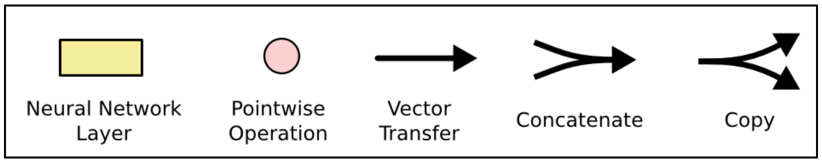
From RNN to LSTM

Simple recurrent neural networks have long-term memory in the form of weights. The weights change slowly during training, encoding general knowledge about the data.They also have short-term memory in the form of ephemeral ac- tivations, which pass from each node to successive nodes.

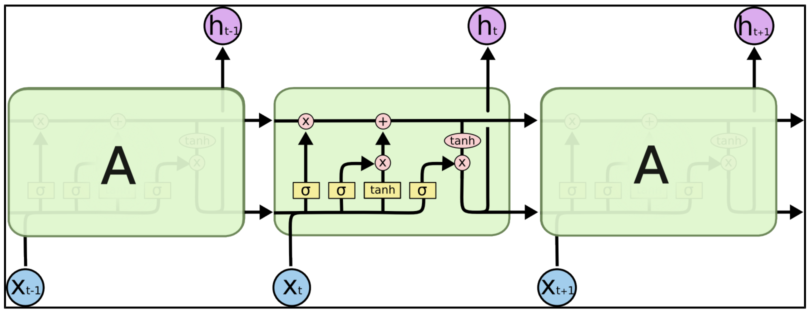
This model resembles a standard recurrent neural network with a hidden layer, but each ordinary node in the hidden layer is replaced by a memory cell.To distinguish references to a memory cell and not an ordinary node, we use the subscript C.

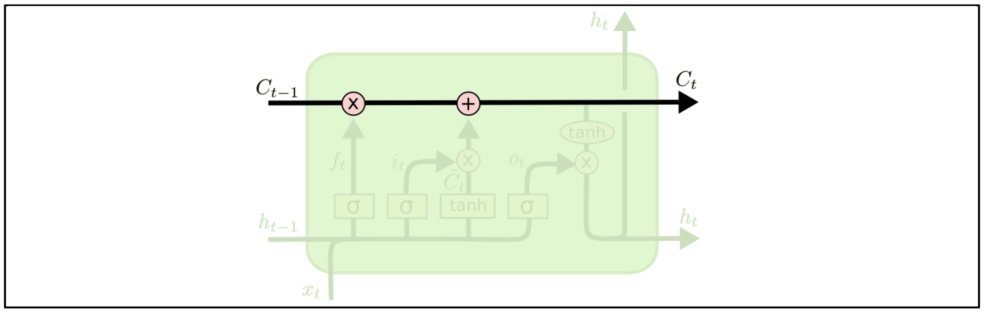
LSTM cell.

Legend;

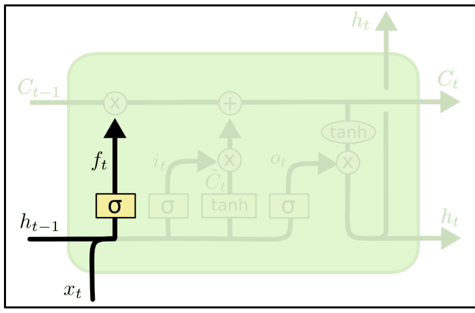


General scheme;



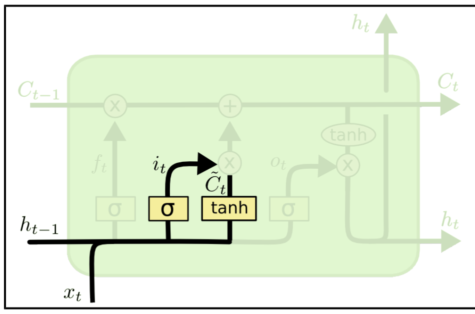


The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions.The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks ath$_{t1}$andx$_t$, and outputs a number between0and1for each number in the cell stateC$_{t-1}$.More details:f$_t$=(W$_{fx}$* x $_t$ + W$_{fh}$* h$_{t-1}$ + b$_f$);Another:f$_t$=(W$_{fx}$* x$_t$+U$_{fh}$* h$_{t-1}$) *a new input vectorx$_t$(including the bias term);

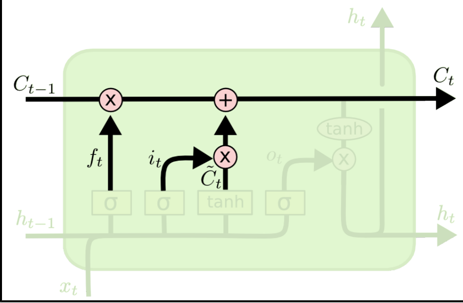


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C'$_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

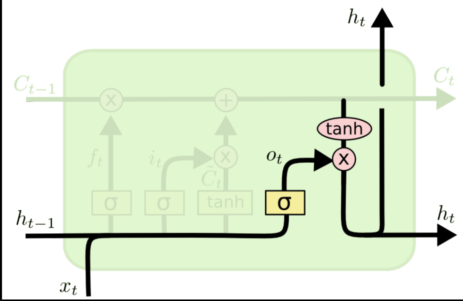
More details (input gate):i$_t$=(W$_{ix}$* x$_t$+ W$_{ih}$* h$_{t-1}$+ b$_i$);Another (input gate):i$_t$=(W$_{ix}$* x$_t$+U$_{ih}$* h$_{t-1}$);

More details (input node /new candidate values / g$_t$/ a$_t$/C'$_t$ /g$_c$):g$_t$=tanh(W$_{gx}$* x$_t$+ W$_{gh}$* h$_{t-1}$+ b$_g$);Another (input node):a$_t$=tanh(W$_{Cx}$* x$_t$+U$_{Ch}$* h$_{t-1}$);



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

It's now time to update the old cell state,C_{t1}, into the new cell stateC_t. The previous steps already decided what to do, we just need to actually do it.We multiply the old state byf_t, forgetting the things we decided to forget earlier. Then we add$i_t C'_t$. This is the new candidate values, scaled by how much we decided to update each state value. More details: at the heart of each memory cell is a node sc with linear activation, which is referred to in the original paper as the "internalstate" of the cell. The internal state has a self-connected recurrent edge with fixed unit weight. Because this edge spans adjacent time steps with constant weight, error can flow across time steps without vanishing or exploding. This edge is often called the constant error carousel.

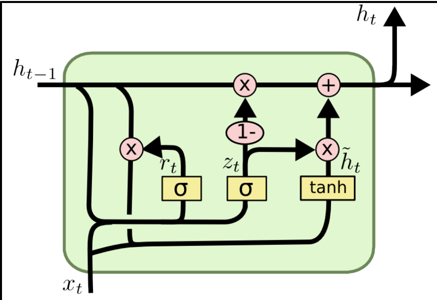


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

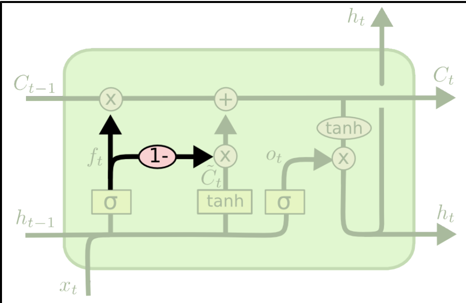
Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state throughtanh(to push the values to be between1and1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.More details: o_t=(Wo$_x$* x_t+ Wo$_h$* h_{t-1}+ bo);Another: o_t=(Wo$_x$* x_t+Uo$_h$* h_{t-1});

Типы LSTM.

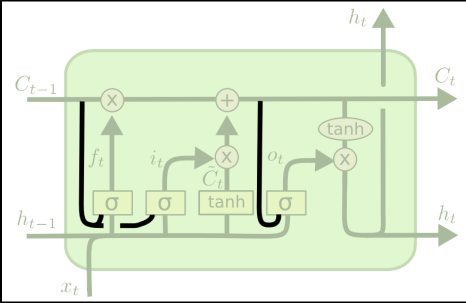
The implementation of BLSTM layer is detailed descripted inAlex Graves. 2012. Supervised sequencelabelling with recurrent neural networks, volume385. Springer.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



$$C_t = f_t * C_{t-1} + \mathbf{(1 - f_t)} * \tilde{C}_t$$



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Peephole connections.

Dimensions;

If the inputx$_t$is of sizenX1and we havedmemory cells, then the size of each ofW$_{*x}$andU$_{*h}$isdXn,andd Xd;

Ignoring the non-linearities,

$$z^t = \begin{bmatrix} \hat{a}^t \\ \hat{i}^t \\ \hat{f}^t \\ \hat{o}^t \end{bmatrix} = \begin{bmatrix} W^c & U^c \\ W^i & U^i \\ W^f & U^f \\ W^o & U^o \end{bmatrix} \times \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}$$
$$= W \times I^t$$

The size ofWwill then be4dX (n+d).Note that each one of thedmemory cells has its own weightsW$_{*x}$andU$_{*h;}$

From LSTM to BILSTM:

1. very effectivefor tagging sequential data.
2. Long Short-Term Memory networks are the same as RNNs,except that the hidden layer updates are replacedby purpose-built memory cells. As a result, theymay be better at finding and exploiting long rangedependencies in the data.
3. A bidirectionalLSTM (BLSTM)furthermore, introduces two independent layers toaccumulate contextual information from the pastand future histories.

References:

1. Zachary C. Lipton,John Berkowitz,Charles Elkan.A Critical Review of Recurrent Neural Networksfor Sequence Learning. 2015.
2. http://colah.github.io/
3. http://arunmallya.github.io/

Интересные примеры применения данного типа сети для решения различных лингвистических задач

Задача	Публикация	Данные	Признаки	Настройки сети
Transliteration	https://yerevann.github.io/2016/09/09/automatic-transliteration-with-lstm/	• We chose Armenian Wikipedia as the easiest available large corpus of Armenian text; • To generate the input sequences for the network we need to romanize the texts. We use probabilistic rules, as different people prefer different romanizations. Why -> the same Latin character can correspond to different Armenian letters; The romanization rules vary a lot in different countries; Armenian language has two branches: Eastern and Western Armenian. These branches have crucial differences in romanization rules; • When generating a chunk to give to the system we drop the ones that do not contain at least 33% Armenian characters; • All symbols are encoded as one-hot vectors and are passed to the network. In our case the input vectors are 72 dimensional and the output vectors are 152 dimensional; • we decided to explicitly align the Armenian sequence by adding some placeholder symbols after those characters that are romanized to multi-character Latin;	character level data;	• bidirectional LSTM networks (BI-LSTM); • We have also added a shortcut connection from the input t the output of the 2nd biLSTM layer. This should help to lear the “easy” transliteration rules on this short way and leave LSTMs for the complex stuff. • adagrad algorithm; • learning rate was set to 0.01; • batch_size 350;

Morphological Segmentation	L. Wang, Z. Cao, Y. Xia, G. de Melo. Morphological Segmentation with Window LSTM Neural Networks. 2016	This well-known data set by Snyder and Barzilay (2008) was derived from biblical text and consists of Hebrew, Arabic, Aramaic, and English terms together withtheir frequencies. The morphologically segmented Hebrew and Arabic words, of which there are 6,192 each.	one-hot vector предыдущих, последующих (по отношению к ”центральному” символу) символов в ”окне”.	• 0.25 as the dropout to apply right after the encoder to an LSTM in the Window LSTM architecture; • dropout as 0.3 and learning rate to be 0.0005 for Multi-Win LSTMs (MW-LSTM) architecture; • dropout as 0.2, learning rate as 0.00065, gradient clipping threshold at 10, decay rate as 0.5, momentum as 0.01 for Bidirectional Multi-Window LSTMs • (BMW-LSTM). Window sizes were chosen from {3,5,7};
POS-tagging	A. Popov. Deep Learning Architecture for Part-of-Speech Tagging with Word and Suffix Embeddings. 2016	BulTreeBank corpus. 38000 sentences with POS tagged words. 3500 of those were used for validation purposes and another 3500 were used for testing. The rest (30000) were used as training data. The tagset used for training and testing has 153 labels.	• word embeddigs. Skip-gram Word2Vec model distributed with TensorFlow;the dimensionality of the vectors was set to 200; a corpus of Bulgarian texts was compiled, consisting of roughly 220 million words (including most of the Bulgarian Wikipedia, news articles and literary texts). It contains about 457000 unique frequent words (the total number of unique words is about 1.6 million; words are considered frequent if they occur at least 5 times in the corpus); • suffix embeddings. 10 million words was extracted from the Bulgarian Wikipedia. Skipgram. There are about 20000 unique frequent suffixes in the training corpus, for which vector representations have been calculated (the total number of unique suffixes is about 37000). The dimensionality of the suffix embeddings is set to 50.	bidirectional LSTM. a vanilla version of the implementation was us That is, the network has only one hidden layer (which contains the two LSTM cells, each of them in turn containing four hidden layers does not make use of popular optimization tricks such as adding regularization, momentum, dropout. The learning rate is set to 0.3 for both setups; for the first one the hidden layers in the LSTMs are initialized with 100 neurons, while the second one the initialization is with 125 neurons;

	Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. 2015 (Baidu research)	Penn TreeBank (PTB);	• spelling features (whether start with a capital letter; whether has all capital letters; letters only, for example, I. B. M. to IBM; etc. (~12)); • context features (unigram, bi-gram and tri-gram features); • word embedding (Senna embedding, 130K vocabulary size and each word corresponds to a 50-dimensional embedding vector);	LSTM networks, bidirectional LSTM networks (BI-LSTM), LSTM networks with a CRF layer (LSTM-CRF), and bidirectional LSTM networks with a CRF layer (BI-LSTM-CRF).
	P. Wang, Y. Qian, F. K. Soong, L. He, H. Zhao. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. 2015. (Microsoft Research Asian)	the Wall Street Journal data from Penn Treebank III;	• word embedding without using morphological features (special algorithm for training, corpora North American news); • full lowercase, full uppercase or leading with a capital letter;	bidirectional LSTM networks (BI-LSTM); исп. библиотека: https://s ceforge.net/projects/currennt/ http://www.jmlr.org/papers/volume16/weninger15a/weninger15a the input layer = 100, output layer = size is set as the number of t types according to the specific tagging task, hidden layer (мона 1) 300 (мона 100); decoder: a special transition matrix A between each step's output;
Chunking	Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. 2015 (Baidu research)	CoNLL 2000;	Z	Z
	P. Wang, Y. Qian, F. K. Soong, L. He, H. Zhao. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. 2015. (Microsoft Research Asian)	CoNLL 2000;	Z	Z
NER	Z. Huang, W. Xu, K. Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. 2015 (Baidu research)	CoNLL 2003;	Z	Z
	P. Wang, Y. Qian, F. K. Soong, L. He, H. Zhao. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. 2015. (Microsoft Research Asian)	CoNLL 2003;	Z	Z