Using Context Features for Morphological Analysis of Russian

Alexey Sorokin[1,2]

[2]Moscow Institute of Science and Technology, [1]Moscow State University

"Dialogue", International Conference
on Computational Linguistics,
Moscow, June, 2st, 2018

Morphological tagging: problem setting

- Morphological tagging determines morphological labels of words in text:

Его	DET
решение	NOUN, case=Nom, gender=Neut, number=Sing
задачи	NOUN, case=Gen, gender=Fem, number=Sing
было	AUX, mood=Ind, tense=Past, aspect=Imp gender=Neut, number=Sing
неправильным	ADJ, case=Ins, gender=Neut, number=Sing

Morphological tagging: problem setting

- Morphological tagging determines morphological labels of words in text:

Его	DET
решение	NOUN, case=Nom, gender=Neut, number=Sing
задачи	NOUN, case=Gen, gender=Fem, number=Sing
было	AUX, mood=Ind, tense=Past, aspect=Imp gender=Neut, number=Sing
неправильным	ADJ, case=Ins, gender=Neut, number=Sing

- Applications:
  - Downstream text processing: syntactic parsing, named entity recognition.
  - Features for more complex NLP tasks.
  - Automatic corpora annotation.

Morphological tagging: example

Word	Num. of tags	Tags
Его	5	PRON, gender=Masc, case=Gen PRON, gender=Masc, case=Acc PRON, gender=Neut, case=Gen PRON, gender=Neut, case=Acc **DET**
решение	2	**NOUN, case=Nom** NOUN, case=Acc
задачи	3	**NOUN, number=Sing, case=Gen** NOUN, number=Plur, case=Nom NOUN, number=Plur, case=Acc
было	2	**AUX** PART
неправильным	3	**ADJ, number=Sing, gender=Masc, case=Ins** ADJ, number=Sing, gender=Neut, case=Ins ADJ, number=Plur, case=Dat
	180	

Morphological tagging: approaches

- Most algorithms search for the most probable tags:

$$\widehat{\mathbf{t}} = \text{argmax}_{\mathbf{t}}\, p(\mathbf{t}|\mathbf{w})$$

- Classical approach to morphological tagging: hidden Markov models.

$$\begin{aligned} p(\mathbf{t}|\mathbf{w}) &= p(\mathbf{w}|\mathbf{t})p(\mathbf{t}) \\ p(\mathbf{t}) &- \text{ngram model} \\ p(\mathbf{w}|\mathbf{t}) &= p(w_1|t_1)\ldots p(w_m|t_m) \end{aligned}$$

Morphological tagging: approaches

- Most algorithms search for the most probable tags:

$$\widehat{\mathbf{t}} = \text{argmax}_{\mathbf{t}}\, p(\mathbf{t}|\mathbf{w})$$

- Classical approach to morphological tagging: hidden Markov models.

$$\begin{aligned} p(\mathbf{t}|\mathbf{w}) &= p(\mathbf{w}|\mathbf{t})p(\mathbf{t}) \\ p(\mathbf{t}) &— \text{ngram model} \\ p(\mathbf{w}|\mathbf{t}) &= p(w_1|t_1)\ldots p(w_m|t_m) \end{aligned}$$

- ngram model tries to recover local dependencies between tags (e. g. nominal phrase agreement).

Morphological tagging: approaches

- Problems with HMMs:
  - Not all dependencies are local.
  - Lexical selectional preferencies are not encoded:

нет возможности	*there is no possibility*+Sg+Gen
есть возможности	*there are possibilities*+Pl+Nom

  - Unreliable tag probabilities for unknown words.

Morphological tagging: approaches

- Problems with HMMs:
  - Not all dependencies are local.
  - Lexical selectional preferencies are not encoded:

нет возможности	*there is no possibility*+Sg+Gen
есть возможности	*there are possibilities*+Pl+Nom

  - Unreliable tag probabilities for unknown words.
- HMMs model local dependencies between tags, but not dependencies between words and tags.

Conditional random fields

- CRF probability formula:

$$
\begin{aligned}
p(\mathbf{t}|\mathbf{w}) &= \prod_{i=1}^{n} p(t_i|t_{i-1}, \mathbf{w}) \\
p(t_i|t_{i-1}, \mathbf{w}) &= \exp \sum_k \theta_k f_k(t_{i-1}, t_i, \mathbf{w})
\end{aligned}
$$

- f_k – features, encoding dependencies between adjacent tags and words in immediate context.

Conditional random fields

- CRF probability formula:

$$
\begin{aligned}
p(\mathbf{t}|\mathbf{w}) &= \prod_{i=1}^{n} p(t_i|t_{i-1}, \mathbf{w}) \\
p(t_i|t_{i-1}, \mathbf{w}) &= \exp \sum_k \theta_k f_k(t_{i-1}, t_i, \mathbf{w})
\end{aligned}
$$

- f_k – features, encoding dependencies between adjacent tags and words in immediate context.
- Features include:
  - Words themselves.
  - Morphological tags and features (case, gender, etc.)
  - Combinations of adjacent tags and words.

CRF properties

- CRF model local dependencies between words and tags.
- Are not so good in modelling global dependencies.

CRF properties

- CRF model local dependencies between words and tags.
- Are not so good in modelling global dependencies.
- Features are hand-engineered.
- The number of features is enormous (several millions), which complicates their learning.
- For morphologically complex languages consumes lot of memory and time without careful management of learning process.

Neural morphological tagging

- Neural networks directly model the probability $p(\mathbf{t}|\mathbf{w})$.
- Basic architecture [Heigold, 2017]: all tags are independent

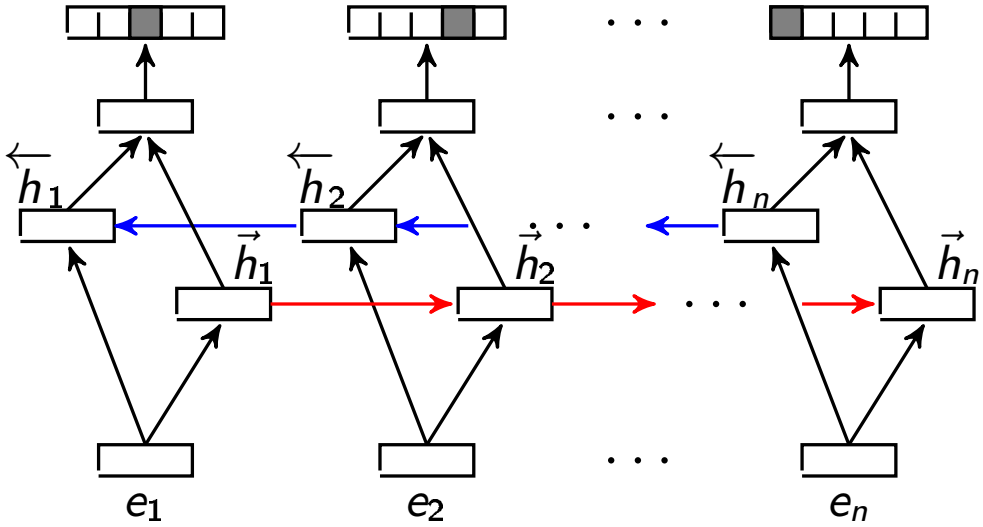
$$p(\mathbf{t}|\mathbf{w}) = \prod_i p(t_i|\mathbf{w})$$

Neural morphological tagging

- Neural networks directly model the probability $p(\mathbf{t}|\mathbf{w})$.
- Basic architecture [Heigold, 2017]: all tags are independent

$$p(\mathbf{t}|\mathbf{w}) = \prod_i p(t_i|\mathbf{w})$$

- Tags are predicted using recurrent network (usually BiLSTM):



Neural morphological tagging

- Neural network consists of two parts:
  - Character-level network for word embeddings.
  - Word-level network, transforming word embeddings $e_1, \ldots, e_n$ to probability distributions $\pi_1 = p(t_1|e_1), \ldots, \pi_n$.

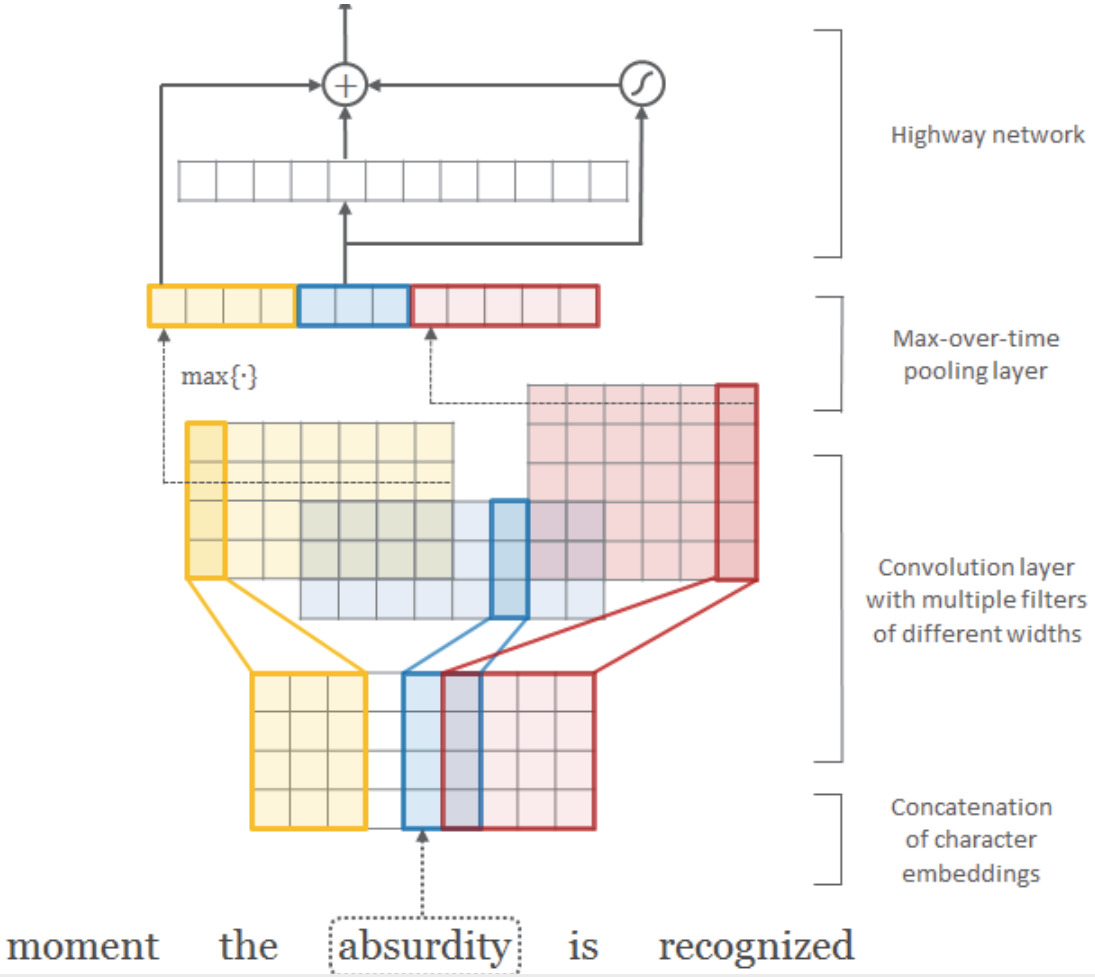
Neural morphological tagging

- Neural network consists of two parts:
  - Character-level network for word embeddings.
  - Word-level network, transforming word embeddings $e_1, \ldots, e_n$ to probability distributions $\pi_1 = p(t_1|e_1), \ldots, \pi_n$.
- Word-level network is a bidirectional LSTM:

$$\begin{aligned}
\vec{h}_i &= LSTM(\vec{h}_{i-1}, e_i),\ i = 1, \ldots, n \\
\overleftarrow{h}_i &= LSTM(\overleftarrow{h}_{i+1}, e_i),\ i = n, \ldots, 1 \\
z_i &= W[\vec{h}_i, \overleftarrow{h}_i] + b,\ i = 1, \ldots, n \\
\pi_i &= \mathrm{softmax}(z_i)
\end{aligned}$$

Word embeddings

- Word embeddings are constructed using architecture from [Kim et al., 2015]:



Baseline architecture: drawbacks

- LSTM effectively extracts relevant information from the context to predict morphological tags.
- But it does not model tag interactions!

Baseline architecture: drawbacks

- LSTM effectively extracts relevant information from the context to predict morphological tags.
- But it does not model tag interactions!

Я шёл по+Dat затерянной в садах узкой+Dat улице+Dat

~~узкой+Loc~~ ~~улице+Loc~~

Baseline architecture: drawbacks

```
x::я шел по затерянной в садах узкой улице
>> 1    я          PRON    Case=Nom|Number=Sing|Person=1
2       шел        VERB    Aspect=Imp|Gender=Masc|Mood=Ind|Number=Sing|Te
3       по         ADP     _
4       затерянной         VERB    Aspect=Perf|Case=Dat|Gender=Fem|Number
5       в          ADP     _
6       садах      NOUN    Animacy=Inan|Case=Loc|Gender=Masc|Number=Plur
7       узкой      ADJ     Case=Dat|Degree=Pos|Gender=Fem|Number=Sing
8       улице      NOUN    Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
```

- The trigger for Dative case is relatively remote (the preposition *по*).
- The system may fail to predict the case of adjective *узкой*, but correctly recognize the case of noun *улице*.
- It has no ability to check their agreement.

Language model on tags: motivation

- The system should be able to discriminate between probable and improbable sequences of tags.
- This is what a language model does (ngram model in HMMs).
- We apply a neural language model on tags to deal with long-distance dependencies.

Language model on tags: motivation

- The system should be able to discriminate between probable and improbable sequences of tags.
- This is what a language model does (ngram model in HMMs).
- We apply a neural language model on tags to deal with long-distance dependencies.
- Tags are encoded as 0/1-vectors with ones corresponding to features:

NOUN, case=Gen, gender=Fem, number=Sing

Noun	Noun, case=Gen	Adj, case=Gen	Noun, gender=Fem	Noun, gender=Masc	Noun, Number=Sing
1	1	0	1	0	1

- They are then embedded using dense matrix and passed through RNN with local attention from [Tran et al., 2016].

CharWeight model

- In each position i of the sentence we have two probaility distributions:

$$\begin{aligned} p_{\text{BASE}}(t_i|\mathbf{w}) &- \text{the basic model} \\ p_{\text{LM}}(t_i|t_1 \dots t_{i-1}) &- \text{language model on tags} \end{aligned}$$

- They are combined using log-linear model (**CharWeight** architecture):

$$\log p(t_i|\mathbf{w}) \sim \sigma p_{\text{BASE}}(t_i|\mathbf{w}) + (1-\sigma) p_{\text{LM}}(t_i|t_1 \dots t_{i-1})$$

CharWeight model

- In each position i of the sentence we have two probaility distributions:

$$\begin{aligned} p_{\mathrm{BASE}}(t_i|\mathbf{w}) &- \text{the basic model} \\ p_{\mathrm{LM}}(t_i|t_1 \dots t_{i-1}) &- \text{language model on tags} \end{aligned}$$

- They are combined using log-linear model (**CharWeight** architecture):

$$\log p(t_i|\mathbf{w}) \sim \sigma p_{\mathrm{BASE}}(t_i|\mathbf{w}) + (1-\sigma) p_{\mathrm{LM}}(t_i|t_1 \dots t_{i-1})$$

- Weight σ is calculated using perceptron:

$$\begin{aligned} \sigma &= \mathrm{sigmoid}(W[s_{i,\mathrm{BASE}}, s_{i,\mathrm{LM}} pos_i] + b) \\ s_{i,\mathrm{BASE}} &- \text{base model LSTM state} \\ s_{i,\mathrm{LM}} &- \text{language model LSTM state} \\ pos_i &= \log(1+i) \text{ (position encoding)} \end{aligned}$$

CharFusion model

- Model weighting works when at least one model is reliable.
- If both fail, it does not help.
- CharFusion approach: concatenate output states of both model and pass them through multilayer perceptron.

$$
\begin{aligned}
s_i &= [s_{i,\mathrm{BASE}}, s_{i,\mathrm{LM}}, pos_i], \\
z_i^{(1)} &= ReLU(W^{(1)} s_i + b^{(1)}), \\
z_i^{(2)} &= W^{(2)} z_i^{(1)} + b^{(2)}, \\
\pi_i &= \mathrm{softmax}(z_i^{(2)})
\end{aligned}
$$

Model parameters

- Character-level network:
  - Symbol embedding size: 32.
  - Convolutional windows widths: from 1 to 7.
  - Filters: $\max(200, 50 \cdot \mathrm{width})$.
  - Total word embedding size: 1100.
  - Convolutional layers: 1.
  - Highway layers: 1.

Model parameters

- Character-level network:
  - Symbol embedding size: 32.
  - Convolutional windows widths: from 1 to 7.
  - Filters: $\max(200, 50 \cdot \mathrm{width})$.
  - Total word embedding size: 1100.
  - Convolutional layers: 1.
  - Highway layers: 1.
- Word-level network:
  - BiLSTM units: 256 (128 in each direction).
  - BiLSTM layers: 1.
  - Dropout rate: 0.2 (0.3 on small datasets).

Model parameters: language model

- Language model parameters:
  - Input emdedding size: 96.
  - Local context width: 5.
  - LSTM units: 128, lstm layers: 2 (before and after local attention).
  - Dropout rate: 0.2 (0.3 on small datasets).

Model parameters: language model

- Language model parameters:
  - Input emdedding size: 96.
  - Local context width: 5.
  - LSTM units: 128, lstm layers: 2 (before and after local attention).
  - Dropout rate: 0.2 (0.3 on small datasets).
- Combination parameters:
  - Tags LM is trained on the same training dataset.
  - Language model states are dropped with rate 0.2 (0.3 on small datasets).
- Training parameters:
  - Implementation: Keras + Tensorflow.
  - Optimizer: Adam with default parameters and gradient clipping threshold 5.0.
  - Batch size: 32.
  - Training for 75 epochs with early stopping patience 10.

Results: Russian

Results on UD2.0 ru_syntagrus dataset (train part for training, dev for validation, test for evaluation).

	[Heigold, 2017]	CharWeight	CharFusion
Tag accuracy (train)	95.19	95.54	95.70
Tag accuracy (test)	95.22	95.52	95.70
Sent. accuracy (train)	52.22	54.95	57.15
Sent. accuracy (test)	50.98	53.66	56.29

Results: other languages

Language	# sent.	# tags	[Heigold, 2017]	CharFusion
Czech (cltt)	465	364	89.28 28.94	89.20 28.30
Turkish	3685	975	87.78 37.54	87.69 37.13
Polish	6100	994	84.82 31.55	86.28 35.91
Arabic	6176	352	90.77 18.38	91.48 21.91
German	14118	686	83.48 15.05	84.83 21.60

- Problems on small datasets (the baseline model is too weak to provide reliable tags in the beginning of the sentence).
- Gains depend on morphological complexity and data amount.

Results: Russian+dictionaries

- For each word we extract its possible tags obtained with PyMorphy.
- These tags are automatically decoded to UD2.0 format (probably with errors).
- A 0/1 vector with all possible UD2.0 tags is embedded by dense matrix and then appended to word embedding.
- Another variant: simply embed source PyMorphy tags.

Model	Tag accuracy	Sent accuracy
[Heigold, 2017]	95.22	50.98
CharFusion	95.70	56.29
[Heigold, 20176]+Pymorphy	96.43	60.01
CharFusion+PyMorphy	96.46	60.70

Results: MorphoRuEval-2017

Model	dev (GICR)	test(VK+Lenta+LJ)
Anastasiev et al., 2017	97.8 NA	97.1 83.3
Sorokin, Yankovskaya, 2017	96.3 78.5	94.8 69.3
Heigold et al., 2017	95.8 74.9	94.6 67.0
Heigold et al., 2017+PyMorphy	96.3 77.4	95.1 68.8
CharFusion	96.1 77.0	94.9 68.0
CharFusion+PyMorphy	96.6 79.8	95.4 71.1

Conclusions

- Integration of language model into morphological tagger improves tagging quality.
- Effect is moderate but consistent ($5 - 10\%$ reduction of error rate) on datasets of 5000 sentences or more.
- On smaller datasets: no or negative effect.
- Further directions:
  - Better language models on tags.
  - Fine-tuning training procedure.
  - Attention architecture [Vaswani, 2017] — attention links mimic syntactic dependencies.
  - Pretrained embeddings, transfer learning...