Improving Part-of-Speech Tagging via Multi-Task Learning and Character-Level Word Representations

Daniil Anastasyev, Ilya Gusev, Eugene Indenbom

ABBYY, Moscow
MIPT, Moscow

Dialogue
24rd International Conference on Computational Linguistics
Moscow, RSUH, 2st June, 2018

Introduction

- The morphological analysis is a key step in many NLP pipelines.

Introduction

- The morphological analysis is a key step in many NLP pipelines.
- The results of morphological analysis are used in syntactic and semantic parsing in ABBYY Compreno.

Introduction

- The morphological analysis is a key step in many NLP pipelines.
- The results of morphological analysis are used in syntactic and semantic parsing in ABBYY Compreno.
- Accurate morphological analyser can highly increase speed of the syntactic parsing by reducing the number of obtained hypotheses.

Task description

Basically, we try to predict the POS tag for each word in the sentence.

У	**двери**	стоял стол секретарши , · · ·
	NOUN Animacy=Inan Case=Gen Gender=Fem Number=Sing	
The interviews took	**place**	two years ago .
	NN	

POS tags' ambiguity

The task cannot be solved without taking the word's context into account:

she	hated	lies
PRP	VBD	NNS
	VBN	VBZ
	JJ	

Work overview

- Every machine learning model relies on the following components:

Work overview

- Every machine learning model relies on the following components:
  1. Data;

Work overview

- Every machine learning model relies on the following components:
  1. Data;
  2. Features extracted from the data;

Work overview

- Every machine learning model relies on the following components:
  1. Data;
  2. Features extracted from the data;
  3. Loss function.

Work overview

- Every machine learning model relies on the following components:
    1. Data;
    2. Features extracted from the data;
    3. Loss function.
- We used BiLSTM POS tagger as a strong baseline.

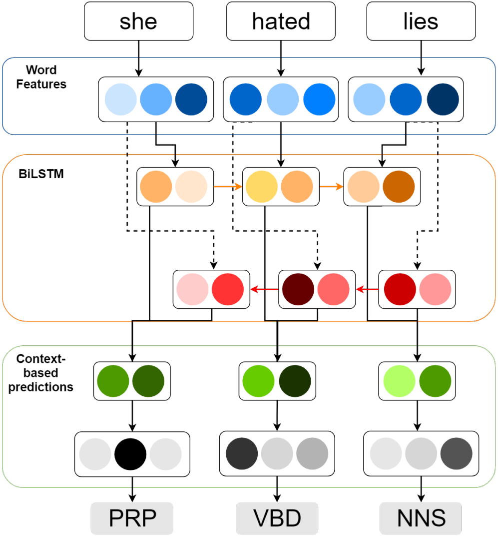
Work overview

- Every machine learning model relies on the following components:
  1. Data;
  2. Features extracted from the data;
  3. Loss function.
- We used BiLSTM POS tagger as a strong baseline.
- We aimed to improve it by changes in these components.

Baseline model

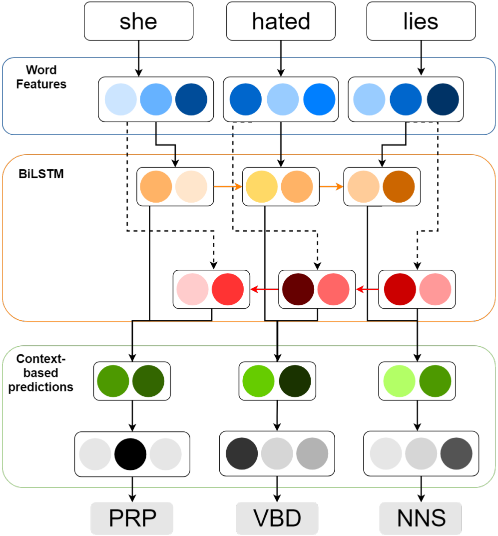
ABBYY

- BiLSTMs are proven to be very effective for POS tagging.



Baseline model

- BiLSTMs are proven to be very effective for POS tagging.
- Tag's prediction is conditioned on the whole sentence.



Result model

In our result model we experimented with:

1. Different types of character-level word embeddings;

Result model

In our result model we experimented with:

1. Different types of character-level word embeddings;
2. Additional grammemes embeddings;

Result model

In our result model we experimented with:

1. Different types of character-level word embeddings;
2. Additional grammemes embeddings;
3. Auxiliary loss functions;

Result model

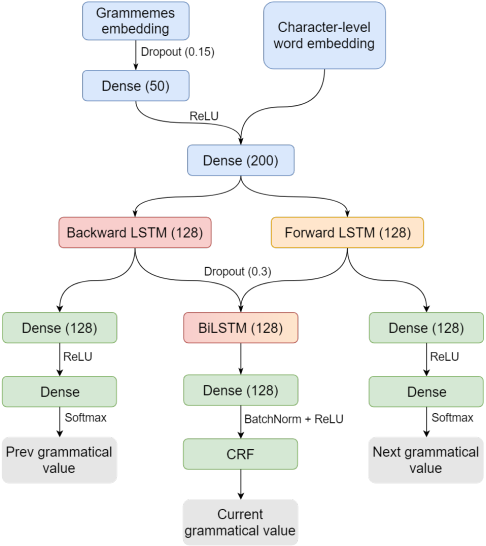
In our result model we experimented with:

1. Different types of character-level word embeddings;
2. Additional grammemes embeddings;
3. Auxiliary loss functions;
4. Extra data for model's pretraining.

Result model

In our result model we experimented with:

1. Different types of character-level word embeddings;
2. Additional grammemes embeddings;
3. Auxiliary loss functions;
4. Extra data for model's pretraining.



Datasets

We compared our results on three datasets:

1. Penn Treebank – English dataset, standard dataset for English models' evaluation;
2. Syntagrus – Russian dataset with Universal Dependencies 2.1 tagset;
3. MorphoRuEval – Russian dataset with Universal Dependencies tagset.

Dataset	Train	Dev	Test	#classes
PTB	912 344	131 768	129 654	45
Syntagrus	871 082	118 630	117 470	721
MorphoRuEval	977 567	108 581	19 560	302

Word embeddings

- Pretrained on a large corpus word vectors contain useful syntactic and semantic relationships.

Word embeddings

- Pretrained on a large corpus word vectors contain useful syntactic and semantic relationships.
- However, word embeddings' matrices are typically very big: 300 dimensional vectors for 50 000 words have 15 000 000 parameters.

Word embeddings

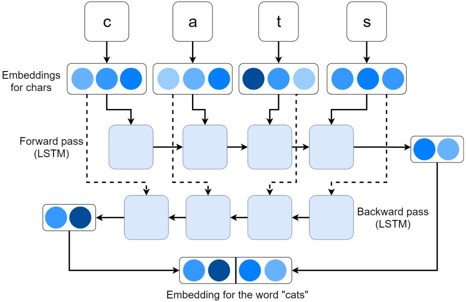
- Pretrained on a large corpus word vectors contain useful syntactic and semantic relationships.
- However, word embeddings' matrices are typically very big: 300 dimensional vectors for 50 000 words have 15 000 000 parameters.
- Another disadvantage of word embeddings is their inability to process out-of-vocabulary words: we can represent them only as a single unknown word vector.

Word embeddings

- Pretrained on a large corpus word vectors contain useful syntactic and semantic relationships.
- However, word embeddings' matrices are typically very big: 300 dimensional vectors for 50 000 words have 15 000 000 parameters.
- Another disadvantage of word embeddings is their inability to process out-of-vocabulary words: we can represent them only as a single unknown word vector.
- To fight these problems we used character-level word embeddings.

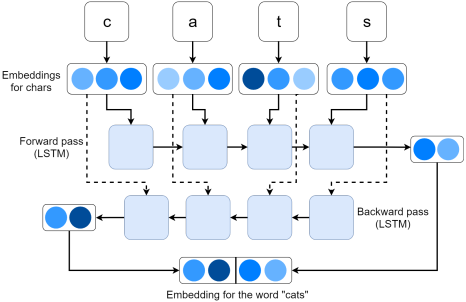
BiLSTM character-level embeddings

- Character-level BiLSTM (Char BiLSTM) is one of the standard ways to build word embeddings.



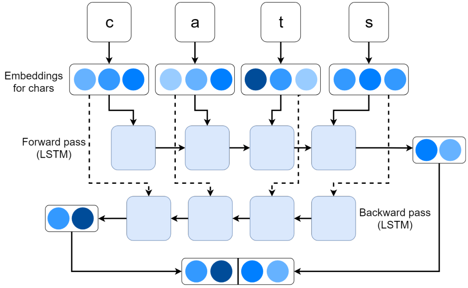
BiLSTM character-level embeddings

- Character-level BiLSTM (Char BiLSTM) is one of the standard ways to build word embeddings.
- Processes characters one by one.



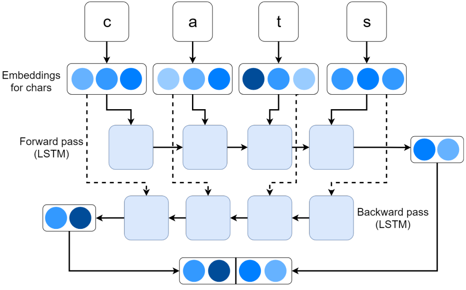
BiLSTM character-level embeddings

- Character-level BiLSTM (Char BiLSTM) is one of the standard ways to build word embeddings.
- Processes characters one by one.
- Handles words with arbitrary lengths.



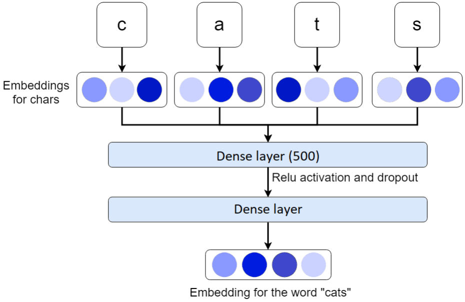
BiLSTM character-level embeddings

- Character-level BiLSTM (Char BiLSTM) is one of the standard ways to build word embeddings.
- Processes characters one by one.
- Handles words with arbitrary lengths.
- Cannot be efficiently parallelized.



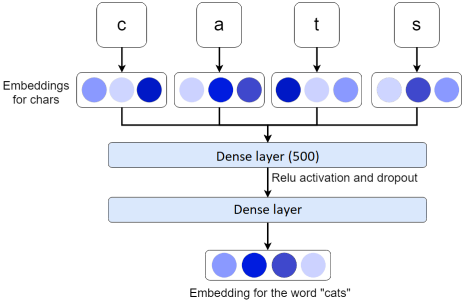
Feed-forward character-level embeddings

- Feed-forward character-level (Char FF) embeddings is our alternative to Char BiLSTM.



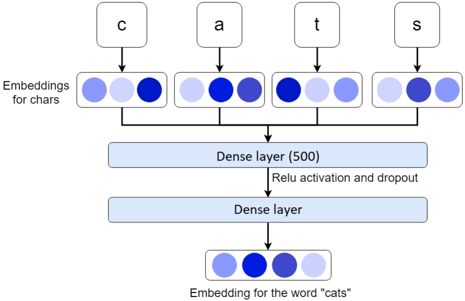
Feed-forward character-level embeddings

- Feed-forward character-level (Char FF) embeddings is our alternative to Char BiLSTM.
- Processes concatenation of characters' embeddings.



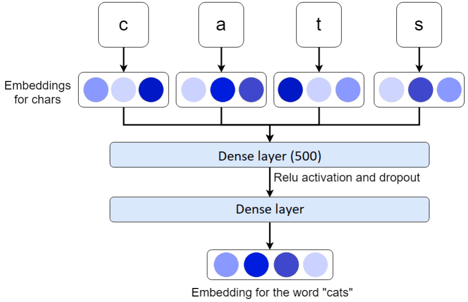
Feed-forward character-level embeddings

- Feed-forward character-level (Char FF) embeddings is our alternative to Char BiLSTM.
- Processes concatenation of characters' embeddings.
- Handles words with fixed lengths (we used 11-13 letters restriction).



Feed-forward character-level embeddings

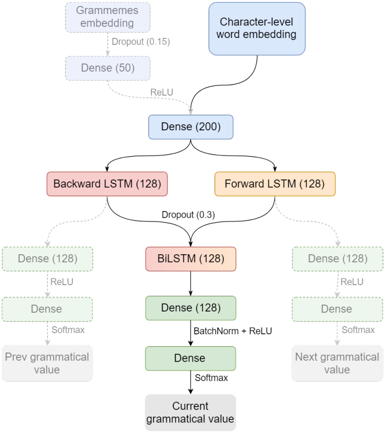
- Feed-forward character-level (Char FF) embeddings is our alternative to Char BiLSTM.
- Processes concatenation of characters' embeddings.
- Handles words with fixed lengths (we used 11-13 letters restriction).
- Can be computed much faster than BiLSTM.



Character-level embeddings comparison

ABBYY

- These two variants of character-level functions obtained approximately equal results.
- However, Char BiLSTM needs twice as many epochs to converge and it works slower.



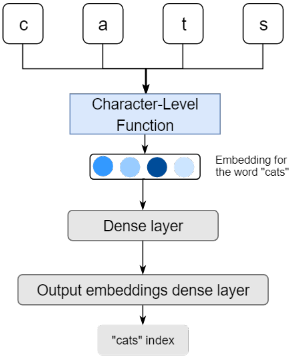
Dataset	Char BiLSTM	Char FF
PTB	97.02% / 96.98%	**97.32%** / **97.26%**
Syntagrus	**95.23%** / **95.39%**	94.98% / 95.16%
MorphoRuEval	96.48% / **94.69%**	**96.68%** / 94.63%

Character-level embeddings' pretraining

- We used pretrained word vectors to incorporate useful information encoded in them into our character-level embeddings.

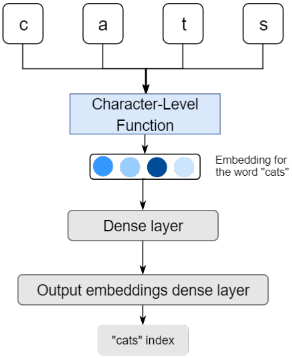
Character-level embeddings' pretraining

- We used pretrained word vectors to incorporate useful information encoded in them into our character-level embeddings.
- We trained an autoencoder-like network to predict word's index by its letters.



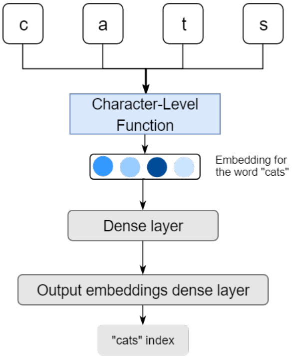
Character-level embeddings' pretraining

- We used pretrained word vectors to incorporate useful information encoded in them into our character-level embeddings.
- We trained an autoencoder-like network to predict word's index by its letters.
- The output layer was initialized by the first 20 thousand pretrained vectors.



Character-level embeddings' pretraining

- We used pretrained word vectors to incorporate useful information encoded in them into our character-level embeddings.
- We trained an autoencoder-like network to predict word's index by its letters.
- The output layer was initialized by the first 20 thousand pretrained vectors.
- The crossentropy loss forced embedding predicted by the character-level function to be similar to the corresponding vector and less similar to all other words' vectors.



Character-level embeddings' pretraining

- The pretrained character-level embeddings were trained further with the whole model.

Character-level embeddings’ pretraining

- The pretrained character-level embeddings were trained further with the whole model.
- The model with pretrained embeddings achieved much higher quality during the first few epochs.

Character-level embeddings' pretraining

- The pretrained character-level embeddings were trained further with the whole model.
- The model with pretrained embeddings achieved much higher quality during the first few epochs.
- The pretraining process led to 4-5% error rate reduction (ERR) on Russian datasets and 2-3% ERR on PTB.

Dataset	Char FF	Char FF (Pretrained)
PTB	97.32% / 97.26%	**97.40%** / **97.31%**
Syntagrus	94.98% / 95.16%	**95.22%** / **95.36%**
MorphoRuEval	96.68% / 94.63%	**96.88%** / **94.63%**

Grammemes embedding

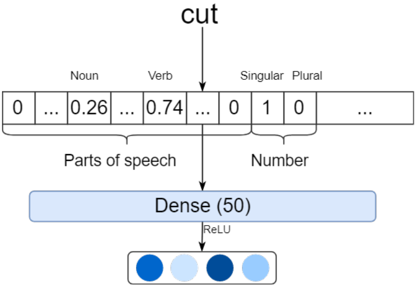
- Word’s form only cannot be a good evidence to its syntactic or semantic value (e.g., «land» vs «laud» or «taxes» vs «takes»).

Grammemes embedding

- Word's form only cannot be a good evidence to its syntactic or semantic value (e.g., «land» vs «laud» or «taxes» vs «takes»).
- We used dictionary information to improve the embeddings' quality.

Grammemes embedding

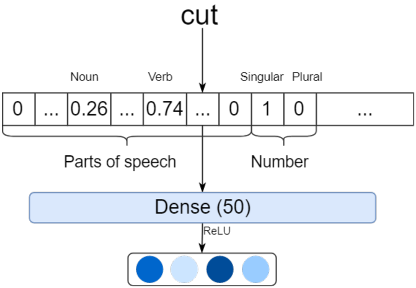
- Word's form only cannot be a good evidence to its syntactic or semantic value (e.g., «land» vs «laud» or «taxes» vs «takes»).
- We used dictionary information to improve the embeddings' quality.
- We estimated the probability of each possible grammeme using the word forms' probabilities.



Grammemes embedding

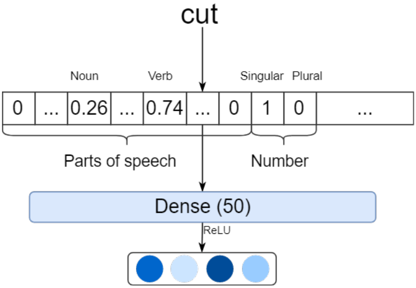
ABBYY

- Word's form only cannot be a good evidence to its syntactic or semantic value (e.g., «land» vs «laud» or «taxes» vs «takes»).
- We used dictionary information to improve the embeddings' quality.
- We estimated the probability of each possible grammeme using the word forms' probabilities.
    - E.g., noun form of the word «cut» has frequency equal to $2.84 \cdot 10^{-5}$, while the verb form has frequency $8.75 \cdot 10^{-5}$. Therefore, $\mathbf{P}(\text{noun}) \approx 0.26$.



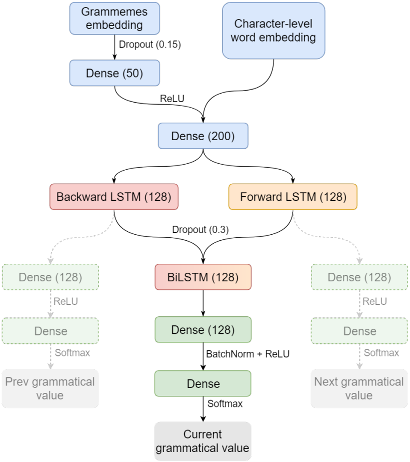
Grammemes embedding

- Word's form only cannot be a good evidence to its syntactic or semantic value (e.g., «land» vs «laud» or «taxes» vs «takes»).
- We used dictionary information to improve the embeddings' quality.
- We estimated the probability of each possible grammeme using the word forms' probabilities.
  - E.g., noun form of the word «cut» has frequency equal to $2.84 \cdot 10^{-5}$, while the verb form has frequency $8.75 \cdot 10^{-5}$. Therefore, $\mathbf{P}(\text{noun}) \approx 0.26$.
- We used an additional dense layer to obtain some relationships between grammemes.



Grammemes embedding

- On Russian datasets the grammemes embeddings gave up to 35% ERR.
- On English dataset the improvement seems marginal.



Dataset	Char FF (Pretrained)	+ Grammemes
PTB	97.40% / **97.31%**	**97.43%** / 97.30%
Syntagrus	95.22% / 95.36%	**96.77%** / **97.00%**
Gikrya	96.88% / 94.63%	**98.07%** / **95.36%**

Multi-task learning

- Multi-task learning is a known way to improve model's quality and make it more robust.

Multi-task learning

- Multi-task learning is a known way to improve model's quality and make it more robust.
- Model is optimized by both main loss function and some auxiliary losses.

Multi-task learning

- Multi-task learning is a known way to improve model's quality and make it more robust.
- Model is optimized by both main loss function and some auxiliary losses.
- It learns to produce more general representations from the data.

Language models' auxiliary losses

- We used language models auxiliary losses to improve the model's quality.

Language models' auxiliary losses

- We used language models auxiliary losses to improve the model's quality.
- *Word language model* additionally tries to predict the next word using Forward LSTM and the previous one with Backward LSTM:

$$\text{Forward LSTM(she, hated)} \sim \begin{matrix} \text{tag(hated)} \\ \text{lies} \end{matrix}$$

Language models' auxiliary losses

- We used language models auxiliary losses to improve the model's quality.
- *Word language model* additionally tries to predict the next word using Forward LSTM and the previous one with Backward LSTM:

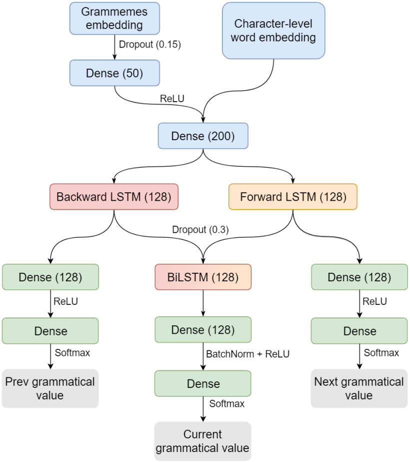
$$\text{Forward LSTM(she, hated)} \sim \begin{matrix}\text{tag(hated)} \\ \text{lies}\end{matrix}$$

- *POS language model* additionally tries to predict the next and the previous words' tags using Forward LSTM and Backward LSTM correspondingly:

$$\text{Forward LSTM(she, hated)} \sim \begin{matrix}\text{tag(hated)} \\ \text{tag(lies)}\end{matrix}$$

Language models' auxiliary losses

- On PTB both loss variants led to equal improvements.
- The POS LM loss gave considerably better results on Russian datasets.
- Error reduction rate on PTB and Syntagrus was about 7-8%, while on MorphoRuEval testset we achieved 36% ERR.



Dataset	Previous Model	+ Word LM	+ POS LM
PTB	97.43% / 97.30%	**97.57% / 97.49%**	**97.57% / 97.49%**
Syntagrus	96.77% / 97.00%	96.69% / 96.96%	**96.97% / 97.24%**
MorphoRuEval	98.07% / 94.85%	97.91% / 96.30%	**98.12% / 96.72%**

CRF output layer

- CRF layer usually helps to improve the quality of sequence labeling models.

CRF output layer

- CRF layer usually helps to improve the quality of sequence labeling models.
- In our case, we were able to achieve a modest improvement only on PTB dataset.

Dataset	Previous Model	+ CRF
PTB	97.57% / 97.49%	**97.60%** / **97.51%**
Syntagrus	**96.97%** / **97.24%**	96.72% / 96.97%
MorphoRuEval	**98.12%** / **96.72%**	98.07% / 96.65%

Transfer learning

- Transfer learning is a popular way to increase model's quality.

Transfer learning

- Transfer learning is a popular way to increase model's quality.
- Usually, we pretrain a model on a large dataset and fine-tune it on a smaller task-specific dataset.

Transfer learning

- Transfer learning is a popular way to increase model's quality.
- Usually, we pretrain a model on a large dataset and fine-tune it on a smaller task-specific dataset.
- We performed transfer learning to Syntagrus from two datasets:

Transfer learning

- Transfer learning is a popular way to increase model's quality.
- Usually, we pretrain a model on a large dataset and fine-tune it on a smaller task-specific dataset.
- We performed transfer learning to Syntagrus from two datasets:
  1. MorphoRuEval dataset with similar UD based tagset;

Transfer learning

- Transfer learning is a popular way to increase model's quality.
- Usually, we pretrain a model on a large dataset and fine-tune it on a smaller task-specific dataset.
- We performed transfer learning to Syntagrus from two datasets:
  1. MorphoRuEval dataset with similar UD based tagset;
  2. Large (10 million tokens) Compreno tagged dataset with different tagset.

Transfer learning

- Transfer learning is a popular way to increase model's quality.
- Usually, we pretrain a model on a large dataset and fine-tune it on a smaller task-specific dataset.
- We performed transfer learning to Syntagrus from two datasets:
  1. MorphoRuEval dataset with similar UD based tagset;
  2. Large (10 million tokens) Compreno tagged dataset with different tagset.
- Both pretrained models achieved approximatelly similar results and showed 38-39.5% ERR in comparison to our best previous model.

Model	Accuracy
Best previous	96.97% / 97.24%
MorphoRuEval pretrained	**98.21%** / **98.33%**
Compreno pretrained	98.18% / 98.29%

Comparison with others results on PTB

- Our result is worse than the best known result on PTB dataset.
- However, this result achieved with a model without word embeddings, which means that our model uses much smaller number of parameters.

Tagger	Test Acc
Manning (2011)	97,32%
Søgaard (2011)	97,50%
Santos (2014)	97,32%
Ling (2015)	**97,78%**
Ma (2016)	97,55%
Choi (2016)	97,64%
Rei (2017)	97,43%
This work	97,51%

Comparison with others results on MorphoRuEval

- Our result is worse than the best known result on MorphoRuEval dataset.
- However, this result is achieved without pretraining and usage of word embeddings.

Tagger	Literature	News	VKontakte
Sorokin, et al	94.16%	93.71%	92.29%
Anastasyev, et al	95.30%	97.54%	95.15%
Anastasyev, with pretrain	**97.45%**	97.37%	**96.52%**
This work	96.46%	**97.97%**	95.64%

Summary

- We proposed a number of improvements to the baseline BiLSTM model.

Summary

- We proposed a number of improvements to the baseline BiLSTM model.
- We describe an alternative character-level function which can be computed faster than BiLSTM and shows better performance.

Summary

- We proposed a number of improvements to the baseline BiLSTM model.
- We describe an alternative character-level function which can be computed faster than BiLSTM and shows better performance.
- We showed a way to pretrain character-level embeddings with standard word embeddings.

Summary

- We proposed a number of improvements to the baseline BiLSTM model.
- We describe an alternative character-level function which can be computed faster than BiLSTM and shows better performance.
- We showed a way to pretrain character-level embeddings with standard word embeddings.
- We introduced a novel POS LM auxiliary loss.

ABBYY

Summary

- We proposed a number of improvements to the baseline BiLSTM model.
- We describe an alternative character-level function which can be computed faster than BiLSTM and shows better performance.
- We showed a way to pretrain character-level embeddings with standard word embeddings.
- We introduced a novel POS LM auxiliary loss.
- We applied transfer learning to highly increase quality of the model.

Summary

- We proposed a number of improvements to the baseline BiLSTM model.
- We describe an alternative character-level function which can be computed faster than BiLSTM and shows better performance.
- We showed a way to pretrain character-level embeddings with standard word embeddings.
- We introduced a novel POS LM auxiliary loss.
- We applied transfer learning to highly increase quality of the model.
- An open-source version of our model is available on https://github.com/IlyaGusev/rnnmorph